IMAGE ACQUISITION AND RECOGNITION FOR ROBOT VISION

by

AJAY TYAGI

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

MAY, 1989

IMAGE ACQUISITION AND RECOGNITION FOR ROBOT VISION

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY

by

AJAY TYAGI

to the

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

MAY, 1989

EE-1989-M-TYA-IMA

8/5/85.

CERTIFICATE

This is to certify that the thesis *IMAGE ACQUISITION AND RECOGNITION FOR ROBOT VISION* has been carried out by *AJAY TYAGI* under our supervision and has not been submitted elsewhere for a degree.

Dr. R. N. BISWAS

Dept. of Electrical Engg.

Dr. H. KARNICK

Dept. of Computer Science & Engg.

Indian Institute of Technology

Kanpur

ACNOWLEDGEMENT

I am greatly indebted to Dr. R.N. Biswas for his perspective and motivation all through the supervision of this thesis.
Due thanks to Dr. H. Karnick for his excellent guidance in software and to Dr. A. Ghosh for his patience and support during the later part of the work.

Thanks are due also for all of my friends and everybody else at the ACES 3rd floor, who made the laboratory environment so congenial. Specially the assistance of Deepak Murthy during the hardware testing was invaluable.
And finally thanks to the hardware I tested, for it made me feel how important a nanosecond is in one's life.

Ajay

(Ajay Tyagi)

14th May '89

ABSTRACT

A framework for a robot vision system to deal with a typical industrial automation problem has been suggested. A frame grabber design has been proposed using PC-AT as the host machine. A new technique for feature extraction and interpretation to deal with the machine parts of complexity found in mechanical assemblies has also been proposed. The technique has been developed modifying the existing Characteristic Views approach. The motivation for proposing such a technique is to use simpler features for 3-D object recognition by employing a moving camera mounted on the robot arm. Occlusion problem has also been addressed but not in its full generality. Hints have been given to develop a new architecture of a real-time computer exploting this technique.

LIST OF FIGURES

LIST OF COMPONENTS

Integrated Circuits

U1	74S133	13 input NAND gate
U2	74LS00	Quad 2-input NAND gate
U3	74LS245	Octal Transceiver
U4	74LS04	Hex Inverter
U5	74LS138	3-8 Decoder
U6	74LS08	Quad 2-input AND gate
U7	74LS75	Quad Bistable latch
U8	74LS245	Octal Transeiver
U9	74LS245	Octal Transceiver
U10	74LS74	Dual J-K Flip Flop
U11	74LS08	Quad 2-input AND gate
U12	74LS00	Quad 2-input NAND gate
U13	74LS74	Dual J-k Flip Flop
U14	74LS08	Quad 2-input AND gate
U15	74LS04	Hex Inverter
U16	74LS74	Dual J-K Flip Flop
U17	74LS08	Quad 2-input AND gate
U18	HM6264	8K Byte RAM
U19	HM6264	8K Byte RAM

Continued ....

U20	HM6264	8K Byte RAM
U21	HM6264	8K Byte RAM
U22	HM6264	8K Byte RAM
U23	HM6264	8K Byte RAM
U24	HM6264	8K Byte RAM
U25	HM6264	8K Byte RAM
U26	74LS155	Dual 2-4 Decoder
U27	74LS191	4-bit Counter
U28	74LS191	4-bit Counter
U29	74LS191	4-bit Counter
U30	74LS191	4-bit Counter
U31	74LS245	Octal Transceiver
U32	74LS245	Octal Transceiver
U33	74LS245	Octal Transceiver
U34	CA3306	6-bit Flash ADC
U35	AD7524	8-bit DAC
U36.	CD4052	Analog Switch
U37	AD517	Op Amp
U38	74LS02	Quad 2-input NOR gate

<u>Capacitor</u>

C1 0.1 uF

<u>**Potentiometer**</u>

R1 10K trim Pot

<u>**Connector**</u>

J1 9 Pin D connector

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Computer Vision deals with the use of computers or other electronic hardware to analyze visual information Thus it addresses the broad problem of extraction and interpretation of information contained in any general scene In this sense it can be compared with human vision which is highly versatile in adapting to different scenes in different environments [1]. Emulation of such versatility in computer vision systems requires tools that are yet to be developed

Present state of the art in computer vision is mainly concerned with its two subareas - Machine Vision and Robot Vision Though there is a considerable overlap in the definitions of the two, the term machine vision is generally used for systems which address very specific industrial automation and few general problems Robot vision is used to cater to the needs of a

more general class of problems usually found in industrial automation based on industrial robots For example automated inspection of silicon wafers in integrated circuits manufacturing is a machine vision problem. Such a system is custom built for this problem and thus is highly inflexible Assembling machine parts coming on a conveyer in random order and with random orientation through an industrial robot is a robot vision problem Such a system is flexible in the sense that incorporating new models, either using existing features or adding to the feature set, will allow new parts to be added to the ensemble and to create new assemblies But systems available presently to deal with this task are still mainly confined to research laboratories only and are not in general in industrial use

Furthermore, machine vision systems have been built to analyze both man-made objects and natural objects (viz trees, rivers, blood cells, tomographs etc) Robot vision systems, however, have been tested for man-made objects only Scenes including man-made objects are easier to analyze, and most of the 3-D vision techniques that have been developed so far are applicable to these domains [2]

The vision problem that has been addressed in this thesis is a typical robot vision problem The task is to identify some industrial objects like screws, washers, nuts, spanners, simple machine parts, subassemblies etc The motivation for choosing this problem comes from the great demand in industry for a system which can identify such objects along with their types, dimensions and orientations etc. to help automate the fabrication, assembly and inspection phases of manufacturing Building a complete working system of

this complexity requires man-hours far beyond those spent on this work. This thesis examines the robot vision problem in the light of the hardware and software required and gives a framework on which such systems can be built

1.2 ORGANISATION OF THE THESIS

A robot vision system consists of three essential components viz acquisition, preprocessing & interpretation Preprocessing is essentially Image Processing which is fairly standard and thus is a matter of details.

Chapter 2 first gives a formal introduction to a robot vision system Based on this introduction it properly specifies the problem being explored in this thesis and proposes a first draft of the design

All feature extraction and interpretation has been done through the simulated silhouette images using Computer Graphics and Projective Geometry on a PC AT. Moving Camera affect has also been simulated As the front and rear ends of a vision system are quite isolated from each other (through preprocessing), the thesis is organised into two parts : one each for the hardware (frame grabber design and implementation) and software (feature extraction and interpretation) These two parts can be read independently.

Chapter 3 starts with a brief overview of video signals, proceeds to choose a computer as the host machine for the vision system and ends by presenting the basic building blocks of a frame grabber based on this machine

Chapter 4 elaborates on the implementation aspects of the frame grabber design suggested in chapter 3. It explains the logic synthesis of each block of the suggested design, emphasises on their interconnections, selects the various chips required and gives the circuit diagram The PCB layout is given in Appendix A 1. This chapter ends with a discussion of the device driver used to make the frame grabber functional as a unit A listing of the code written in 'C' is given in Appendix A 2

Chapter 5 discusses the 3-D object recognition in general, makes a survey of some existing techniques motivates for the technique proposed in this thesis

Chapter 6 proposes a new technique for 3-D object recognition for the class of problems posed in this thesis, justifies its power and limitations and also reports the results of experimentation on some primitive objects using the approach.

As has been mentioned earlier the data for experimentation has been generated by using projective geometry and computer graphics Details of this simulator are given in Appendix A.3

Chapter 7 summarises the work done in hardware and software discusses limitations and indicates directions in which future work needs to be done

CHAPTER 2

BASIC REQUIREMENTS FOR A ROBOT VISION SYSTEM

This chapter proposes the first draft of the design of a robot vision system by first presenting the outline of a general robot vision system in section 2.1 and later in section 2.2 specifying the precise problem addressed in this thesis

2.1 OUTLINE OF A ROBOT VISION SYSTEM

After nearly two decades of work, researchers in Computer Vision have converged to a common belief that it is useless to try and develop a robot vision system that is both cost-effective and general-purpose because vision tasks and, especially their environment can vary enormously [2]. A general structure of a robot vision system that can handle a class of problems cost-

effectively using the existing techniques and those that are yet to be developed is shown in Figure 2 1 This structure consists of six basic building blocks These are *signal transducer, frame memory, signal preprocessor, feature extractor, feature interpreter and robot controller* Furthermore, the system may be Open Loop or Iterative

2.1.1 ***Transducer*** : This converts, relevant object properties (e g surface geometry and reflectance) into a signal The sensors used to produce this kind of a signal are television cameras or other similar devices, but the same processing techniques can be applied to scenes derived from sonar, X-rays, laser scanners, and so on. The signal may be a point (zero-dimensional), a line (one-dimensional), or an image (a two-dimensional array of data points having a well defined spatial organisation) We distinguish between an intensity image, a rectangular array of intensity values as given by a television camera, or X - rays and a range image, a rectangular array of range values as given by a laser range finder

2.1.2 ***Frame Memory*** : The data output rate of the transducer is often higher than the data input and data processing rate of the preprocessor This brings out the need for a frame memory or buffer in between The size of this memory depends upon the type of transducer, the signal preprocessor and the accuracy required in the identification.

2.1.3 ***Preprocessor*** : This is used to improve the signal, in hardware or software, for example removing local image variations by low pass filtering,

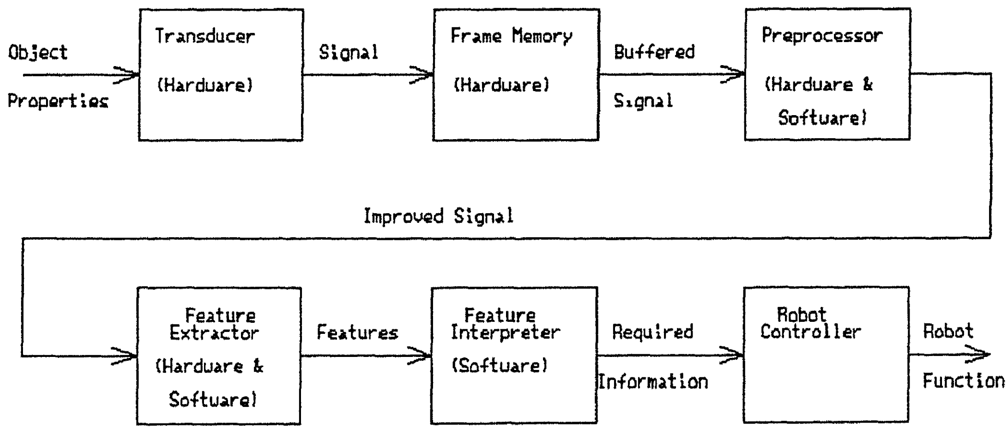

Figure 2.1 COMPONENTS OF A ROBOT VISION SYSTEM

noise removal by frame averaging or smoothing etc Most of these techniques have been derived from image processing which creates an improved image from the original Preprocessing also includes tasks like contrast enhancement, thresholding, edge detection/enhancement etc Often preprocessing consumes an immense amount of time during the identification process Hardware implementations of such procedures become necessary in such cases

2.1.4 ***Feature Extractor*** : This finds object features in hardware and/or software, which may be compared to those of the object models, including the inherent geometric features (e.g average intensity and colour) of the object and its components as well as the relations among these inherent features Making such a description of the object is a harder problem and is what differenciates robot vision from image processing Typical features are areas, moments, lines, curves, shading, texture and the relations among these.

2.1.5 ***Feature Interpreter*** : This matches the extracted features with model features by hypothesis generation and verification, analysing the results, and generating the required information (usually in software) This process involves matching surface edges with those of a model of the object, checking if such a match agrees with other objects and with the laws of physics, and determining the identity and location of the sensed object As mentioned in the previous step, the major features being extracted from the edge-enhanced or contour image of the object are lines, curves, areas, moments, and relations among them Using this description of the object viewed, the task of the

feature interpreter is to find the best match with a similar stored description of the object

2.1.6 Robot Controller : This block receives the interpreted information about the objects viewed and controls the robot arm accordingly. The control of the arm is required either for moving the camera to another location in order to take another view of the objects or the perform some mechanical operation like assembly or inspection of the objects.

2.1.5 Open Loop or Iterative Sensing : The above steps usually coexist in an open-loop forward path Often this scheme of robot sensing fails to generate the required information Iterative sensing is employed in such cases If the interpreted information is insufficient, the vision task program may either try another feature interpretation tool, different features or both Different features may be extracted using either different feature extraction tools, a different improved signal or both A different improved signal may be obtained by applying different preprocessing tools, using a different signal, or both Finally a different signal may be obtained using different transducer configurations.

2.2 PROBLEM SPECIFICATION

The problem of the identification and location of machine parts has attracted the maximum attention in robot vision This task is hard and helps

one gain insight into the powers and limitations of robot vision. The task is also of great practical use in industry

The complexity of the parts to be identified usually varies from simple nuts, bolts, washers, spanners etc to subassemblies like couplings, bearings, bushes etc. Most of the identification work has been carried out under controlled or structured lighting Dealing with arbitrary lighting has proven to be extremely difficult This is not a severe limitation, however, in robot vision as the lighting in an industrial or laboratory environment is fixed and identification process can be calibrated to take that into consideration. Occlusion of one object by another is another major problem in identification This becomes even more complex if the two objects are touching.

The most popular transducer in robot vision is a video camera, usually fixed overhead [3] In a few cases it is supported by additional sensors, proximity switches etc Also there may be more than one camera each fixed at a specified location A recent approach has been to mount the camera on the robot arm itself Identification process is carried out here by taking more than one view of the object. Conversely the camera may still be fixed but the object may be moving on the conveyer belt As the shape information is generally carried in the shading, only monochrome cameras are being used The video signal of the camera is quantised in 256 or more gray levels, but if the lighting is properly designed resulting in relatively better contrast in the video picture, 64 levels are sufficient. The picture is digitised usually in 256x256 format. Higher resolution can be used for better precision but greatly increases the computing requirements Sampling coarser than 256x256

may lead to lack of enough detail

The task of building a framework in this thesis is also oriented towards the identification of machine parts, i.e the estimate of the type and location of the object in a laboratory environment Occlusion problem has also been considered though not in its full generality It is assumed that the picture contains only the object(s) to be identified, i.e. the robot's world consists only of the objects it has been programmed to identify. A standard video camera has been taken as the transducer It should provide a signal of standard 625-B monochrome format though other cameras may also be used with minor modifications in the hardware. Another assumption is that the camera can be mounted on the robot arm Thus more than one view of the object(s) may be available if desired by the decision-making process

2.3 FIRST DRAFT OF THE DESIGN

Based on the specification of the problem as explained in the previous section, the first draft of the robot vision system designed to tackle the problem is shown in Figure 2 2 As is clear from the figure, the analog video signal coming from the camera is digitised and a video field is being stored in the memory of the frame grabber, which also synchronises the digitisation process with the video format. The memory to store a frame (also called as the frame memory) resides in the main memory of the computer or on an external add-on board depending upon the data output rate of the camera and the data input rate of the computer. The video data output rate is determined by the

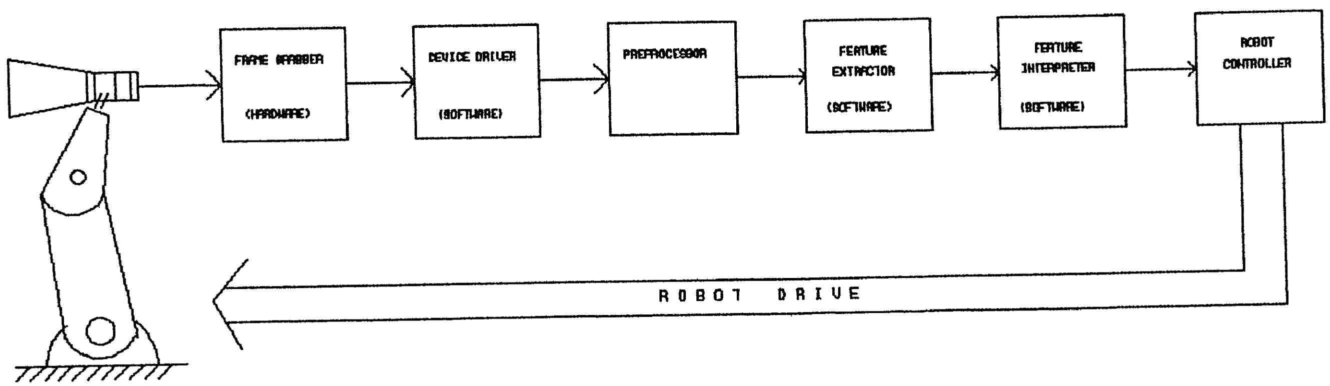

Figure 2.2 ROBOT VISION SYSTEM USING A MOVING CAMERA

standard of the video signal, number of gray levels used to quantise that and the sampling resolution Furthermore the frame grabbing may be in real-time or non real-time Real-time frame grabbing stores the frame at the video rate i e 25 frames/second in 625-B monochrome system However as the software to operate on these stored frames takes much longer time (from few seconds to minutes), most of the frame grabbers are non real-time The data input rate of the computer depends upon the clock speed and architecture of the CPU and its support components

The device driver for the frame grabber issues commands to the frame grabber to capture an image, and either store it in some other block of memory or in the form of a disk file and display the image on a monitor if necessary

Preprocessing is accomplished by calling procedures to operate on the image array. Typical procedures include noise removal through frame averaging or template matching (convolution), thresholding, edge enhancements These procedures take a great deal of time if implemented in software. Hardware implementation is possible through add-on boards (usually based on some form of parallel processing)

The feature description as provided by the feature extractor may contain lists of features one each for each geometrical element like line, circle, arc, ellipse, area, moments etc Each element in the list contains the entity's dimensions and its relationship with other entities in the same or different lists. The procedures to do this constitute the major part of the

software and form a large part of the time required for identification Hardware implementation of such procedures is being pursued but is still in the rudimentary stage due to lack of rigorous theory [4].

The feature interpreter which is always implemented in software is a kind of expert system based on some decision tree or pattern classification approach. In case of occlusion two or more views of the same set of objects may be required for decision making This brings out the need for either having two or more cameras fixed at specified locations or mounting a single camera on the robot arm The later approach is attractive and needs a fast interface between the robot controller and the host computer Through this interface the host computer asks the robot controller to move the robot arm in order to take the camera to a desired location It is assumed that such an interface is available

CHAPTER 3

DESIGN OF THE FRAME GRABBER

This chapter is addressed to the design of the basic building blocks which constitute a frame grabber The chapter begins with a brief outline of the video signal in general and the specifications of the one adopted in India (625 B monochrome) in particular It is assumed that the reader is familiar with the basic concepts of television engineering and composite video signal Section 3 2 is addressed to the hardware requirements of a frame grabber using a suitable computer, enhanced through additional hardware (if found necessary). Section 3 3 concentrates on the choice of various building blocks and their interconnections necessary for such a frame grabber

3.1 VIDEO SIGNAL FORMAT

A television camera or a video camera gives its output signal in a specified format Though this format varies from country to country, almost all of the formats have been derived from three basic formats viz NTSC, PAL and SECAM Furthermore all three of these formats share the same basic structure

Such a structure is called a composite video signal and it consists of a camera signal corresponding to the desired picture information , blanking pulses to make the retrace invisible and synchronising pulses to synchronise the transmitter and receiver scanning [5] A horizontal synchronising pulse (HSYNC) is needed at the end of each active line period whereas a vertical sync pulse (VSYNC) is required after each field is scanned. The amplitude of the HSYNC and VSYNC pulses is kept the same to obtain higher efficiency of the picture signal transmission but their durations (widths) are kept different for separating them at the receiver Since sync pulses are needed consecutively and not simultaneously with the picture signal , these are sent on a time division basis and thus form a part of the composite video signal

3.1.1 ***Interlaced Scanning*** · The scene viewed by the camera is scanned rapidly in both horizontal and vertical directions simultaneously to provide sufficient number of complete pictures or frames per second to give the illusion of continuous motion The frame repetition rate is 25-30 per second in most television systems. However a repetition rate of 50-60 vertical scans per second is employed to reduce the flicker This is accomplished by dividing

each frame into two fields Each field is scanned alternately This method of scanning is known as interlaced scanning

3.1.2 *625 - B Monochrome Format* This is the format adopted in India The basic structure for this format is the same as explained above The specification of this format are given below ·

1 Interlaced scanning at 50 fields/second i.e 25 frames/second

2 Each field consists of 292 5 active lines and 20 lines for vertical retrace, resulting in 625 lines per frame

3 Details of Horizontal Scanning

Period	Time (μs)
Field line	64
Horizontal Sync Pulse	4 5 - 4 9
Front Porch	1 2 - 1.8
Back Porch	5.5 - 6.1
Visible Line Time	51.2-52 8

4 Details of Vertical Blanking

Period	Time (ms)
Total Field Period	20
Visible Field Line	18.7-19 2
Vertical Blanking	0.8 - 1.3

3.2 HARDWARE REQUIREMENTS

In monochrome 625-B system each line has a duration of 52 μs. For a resolution of 256x256 samples there should be 256 samples of the video signal taken per line and 256 such lines. The period of sampling on each line is thus about 0.2 μs, which means the sampling frequency should be 5 MHz. For 256 gray levels each sample will require one byte. One thus requires an 8-bit Analog-to-Digital Converter (ADC) for this purpose. The output will come from the ADC in bursts of 51.2 μs at the rate of 5 Mbytes/second. Each burst will be followed by about 12 μs idle period (of the horizontal retrace). The host computer should be able to sample and store the data at the peak rate i.e. 5 Mbytes/second. Sending this data straight into the main memory of the host computer (with some minor synchronisation logic) may seem to be an attractive way of grabbing the frame. However, if the system does not provide its own shaded graphics monitor, such an approach has a severe limitation as the host system will be busy as long as a frame will be displayed on a separate monitor. Also most of the lower end and intermediate systems are not fast enough to directly grab or display the data at this rate. These requirements bring the need of a dedicated hardware for the frame grabber implemented through TTL logic with onboard counter and 256x256 bytes i.e. 64 Kbytes of RAM to be used as an add-on card for the host machine. While choosing a host computer, these factors should be taken into account along with the cost to performance ratio and the availability of the system.

Commercially available computers range from systems like PC-XT on the lower end to 68020 based workstations on the higher end. Systems like PC-XT

are cheaper but are too slow for robot vision applications. Higher end workstations are attractive for such applications especially from the hardware and software points of view but are costly This suggests an intermediate system like PC-AT as a reasonable compromise to act as a host computer. Its I/O data rate is 2 Mbytes/second only which is reduced even further because of operating system overheads like real-time clock display, memory refresh etc. Therefore, a separate add-on card to act as the frame grabber under the supervision of the PC-AT is a necessity A PC-AT does not support a shaded graphics monitor Thus a separate monitor will also be required The add-on frame grabber card should have logic for driving this monitor also Though such a combination is far from achieving real time performance even if it is equipped with additional hardware to perform preprocessing, it can be built at a cost which is considerably lower than that resulting from a workstation based combination

This add-on hardware should work under the program control of the PC The board should be designed in order to fit in one of the expansion slots of the PC and should communicate with the PC through the PC-bus The PC should be able to give commands to the hardware (to capture a frame for example) and should also be able to access the video frame memory, with its own data input rate In addition the hardware should be able to *display* the stored frame continuously on a monitor if commanded to do so by the PC.

3.3 SPECIFICATIONS FOR THE FRAME GRABBER

It follows from the requirements laid down in the previous section that the frame grabber should possess the following capabilities in addition to the primary capability of capturing a frame

- It should interact with the host PC through the PC-bus available in one of the expansion slots of the PC.
- It should contain a Frame Memory consisting of a 64 KByte-RAM with on board counter and chip select circuitry in addition to the synchronisation circuit
- It should allow the PC to access the Frame Memory as and when required, either for taking the frame for processing into the PC, or for loading a processed frame into the Frame Memory
- It should be able to display the frame stored in its Frame Memory on a monitor as and when asked by the PC
- It should allow PC to access the Frame Memory as and when required
- It should be able to deal with any instability encountered in VSYNC and HSYNC arrival periods without losing synchronism or corrupting the already stored data

Thus the add-on hardware should be able to work in five modes as given as given in Table 3.1 (i) frame capture (FC), (ii) frame display (FD), (iii) frame input (FI), (iv) frame output (FO) and (v) idle or 'do-nothing ' mode (ID). The mode ID is required to reset the various units in the hardware at the boot-up of the PC or while switching between different modes. The mode ID should

override over the other four modes These requirements can be met by using 3 control bits Q_0 Q_1 Q_2 with the bit Q_0 dedicated to the mode ID. The rest two bits can be used to select either of the other four modes

TABLE 3.1

OPERATING MODES OF THE FRAME GRABBER

Mode Name	Mnemonic	Control Bits Q_0 Q_1 Q_2	Operation
Idle	ID	0 X X	Reset and do-nothing
Frame Capture	FC	1 0 0	Capture a frame from the Camera and store it in the Frame Memory
Frame Display	FD	1 0 1	Display the frame stored in the Frame Memory
Frame Input	FI	1 1 0	Transfer a frame from the Frame Memory into the PC Memory
Frame Output	FO	1 1 1	Transfer a frame from the PC Memory to the Frame Memory

3.4 BUILDING BLOCKS OF THE FRAME GRABBER

The standard techniques of digital system design suggest a building block approach as shown in Figure 3.1. The approach has been presented in a fashion that hides the implementation aspects at this stage. This is helpful to modify or upgrade the design for any other host computer. An explanation of the function of the each block follows. The detailed exposition of these blocks and the implementation aspects are covered in the next chapter.

3.4.1 ***Port Address Decoder* (PAD)** This module is used to enable access to the Control Register (for the mode setting), the Status Port (for inquiring the status of the card) and the memory bank (during the mode FI and FO) as and when required by the PC. The Transceiver Set 1 has to be enabled in the proper direction in order to make such accesses. As the usual decoder circuitry do not possess any significant loading to the PC-bus, no separate buffers are needed. However, the IOR line has to be buffered, as it has to control the directions of several transceivers, and IORB represents buffered I/O Read Control.

3.4.2 ***Transceiver Set 1* (TS1)** This Transceiver Set serves the dual purpose of buffering data bus and to act as a bi-directional port for setting the link between the PC and the rest of the frame grabber hardware. These should be enabled on receiving the enable (PA1/PA2) and direction (IORB) signal from the PAD.

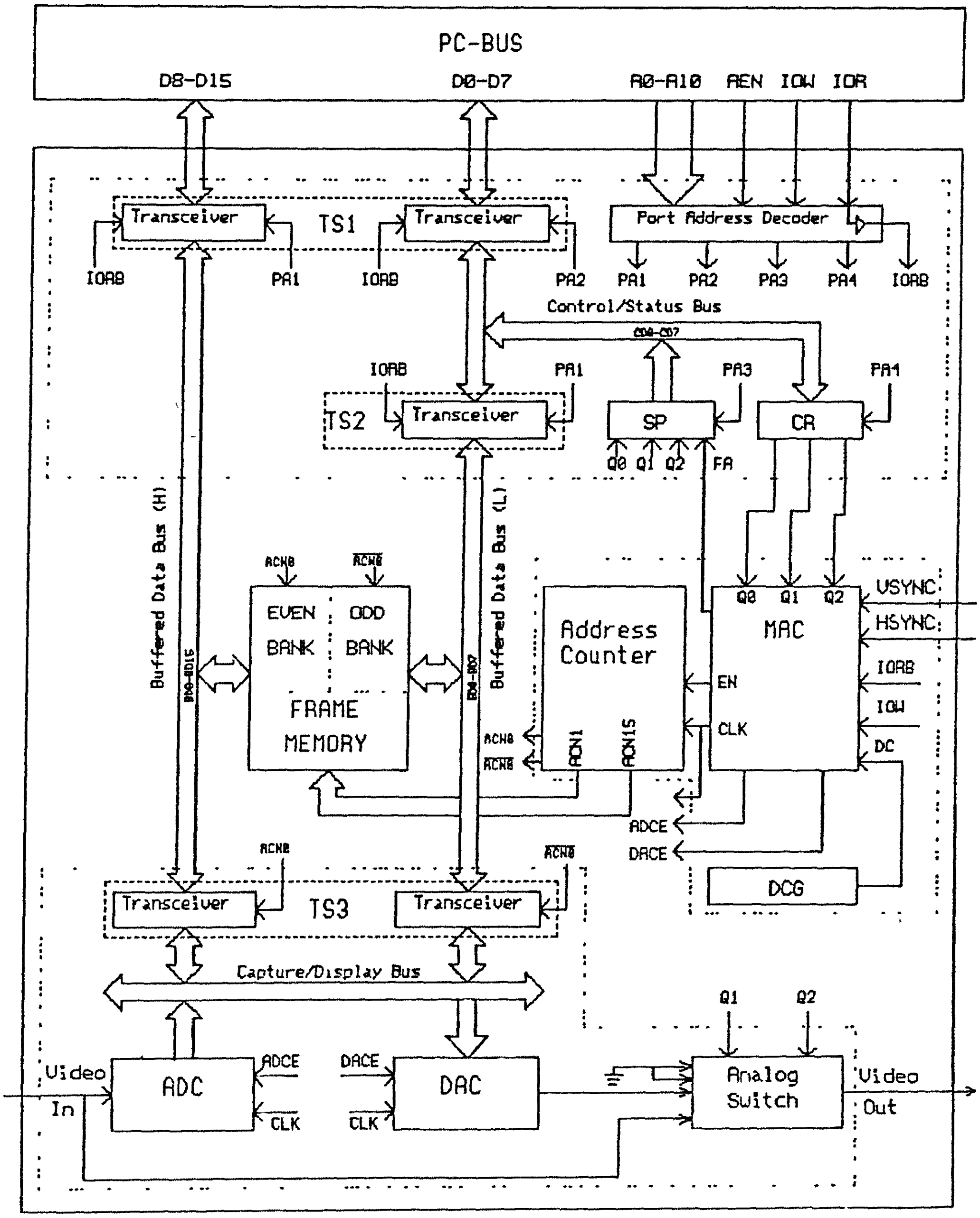

Figure 3.1 FUNCTIONAL BLOCK DIAGRAM OF THE FRAME GRABBER

3.4.3 ***Control Register* (CR)** . This write-only unit receives as input the mode selection code $Q_0Q_1Q_2$ from the PC and makes the code available to the Memory Access Controller which acts as the master controller for the other units The Control Register is enabled by the PA4 output of the PAD and is written into by the PC through the lower half of the data bus.

3.4.4 ***Status Port* (SP)** : This read-only unit sends to the PC the status of the card viz. the current operating mode of the card and information about the status of the frame capture once the CR receives a frame capture command (FC). PC can poll this port till it receives the signal denoting the completion of the frame capture or it can do so after an approximate period of one frame i e. 20 ms It is enabled by the PA3 output of the PAD and is read through the lower half of the data bus.

3.4.5 ***Transceiver Set 2* (TS2)** : This Transceiver is necessary for providing independent access from the PC-bus to the even bank of the Frame Memory on one hand and the CR and SP on the other hand. As both CR and SP are single byte, there is no need to have a transceiver at this level for the upper half of the data bus. The enable for this Transceiver is PA1 Its direction control is set by IORB

3.4.6 ***Frame Memory*** : The Frame Memory is organised in two banks of odd and even addressing This organisation is helpful as the digitised video data coming from the ADC in the mode FC and the data sent to DAC (in the mode FD) comes byte by byte while the PC-AT can access that (in FI and FO) modes word

by word The chip enables for the various modes are generated by decoding the address coming out of the onboard counter

3.4.7 *ADC* : This is an Analog to Digital converter unit used to digitise the camera signal in synchronism with the HSYNC, VSYNC and the dot clock during the mode FC As the video signal is to be digtised at the rate of 5M samples/second this converter should be fast enough to accomplish this Flash ADC on a chip available commercially is necessary for this purpose.

3.4.8 *DAC* · This unit is the Digital to Analog Converter used to generate analog signal for the monitor during the mode FC Again it should be able to convert at the rate of 5M samples/second

3.4.9 *Analog Switch* · This unit is used to select the source of video signal for the monitor depending on the chosen mode While the bypassed camera signal is chosen as this source during the mode FC the source of this signal is switched to the DAC in the mode FD There is no video output signal in the modes FI and FO.

3.4.10 *Transceiver Set 3* (TS3) · This set of Transceivers is necessary (i) to isolate the ADC and the DAC during the modes FI and FO from the data bus on the card and (ii) to channel the ADC output to the proper bank during the mode FC and to multiplex the data going from odd or even bank to the DAC during the mode FD thus preventing shorting of the two halves of the data bus

3.4.11 *Dot Clock Generator* (DCG) : This generates the dot clock of 5 MHz for the MAC, the ADC and the DAC It is usually based on a single dot clock generator chip, but can be built through a simple inverter based logic also.

3.4.12 *Address Counter* (ACN) This 16-bit counter generates the address for the memory in all FOUR modes in which it is accessed. In modes FC and FD it should generate the address in synchronism with the video signal. In mode FI and FO it should generate the address in synchronism with the IOR and IOW signals of the PC. The ACN_0 bit is used for selecting the appropriate bank of the Frame Memory

3.4.13 *Memory Access Controller* (MAC) Its main function is to synchronise the address, data, chip enables and write/read of the frame grabber memory during modes FC and FD with the format of the incoming video signal. It also selects the appropriate clock for the address counter, the ADC and the DAC and controls the chip enables of the ADC and the DAC (ADCE and DACE). It is assumed that the video, HSYNC and VSYNC signals have been separated out from the composite video signal received from the camera by an external analog processing circuit. This makes the card independent of the standard of the signal and also accommodates the cameras which give these signals separately In the mode FC it freezes the card operation (till next ID) after a frame is captured and signals the PC about this through SP In the mode FD it is used to repeat the frame display, by resetting the counter after each frame is displayed. Also the PC is prevented from any access to the Frame Memory during these modes In the modes FC and FD the clock selected is the standard

dot clock of 5 MHz coming from DCG. In other two modes this unit gives the access right for the Frame Memory to the PC In the modes FI and FO the clock is IOR and IOW signals of the PC.

3.1.14 *Internal Buses* . The drivers for the various buses indicated in the Figure 3 1 are shown in Table 3 2 for different modes Also important to note is that the port address for TS2 and the upper half of TS1 are same i.e PA1 The port address for CR is PA4 and for SP is PA3. However, the port address of the lower half of TS1 (PA2) is the logical OR of PA1, PA3 and PA4 This is necessary as this half of TS1 is to be enabled with proper direction for accessing both the CR/SP and the Frame Memory

TABLE 3.2

BUS DRIVERS

Bus	ID	FC	FD	FI	FO
Lower/Upper Data	--	ADC through TS3	Frame Memory	Frame Memory	PC through TS1 &TS2
Capture/Display	--	ADC	Frame Memory through TS3	--	--
Control/Status	PC through TS1	PC through TS1	PC through TS1	Frame Memory through TS2*	PC through TS1*

* The Control/Status Bus is the same as the Buffered Data Bus (Lower) in these modes, and thus carries data which is of no significance for Control/Status information.

CHAPTER 4

IMPLEMENTATION OF THE FRAME GRABBER DESIGN

The design of the frame grabber from the functional point of view has been discussed in the preceding chapter. This chapter focuses on the implementation aspects of the design The actual implementation of each of the conceptual block as shown in Figure 3.1 has to be done keeping in mind its interconnections with other blocks. In order to maintain consistency it is, therefore, convenient to group these blocks in the form of four circuit modules and to discuss the implementation of each of these modules at a time These modules are:

- PC-Bus Interface Module
- Frame Memory
- Capture/Display Module
- System Controller

These four modules are shown by dotted lines in figure 3 1 The PC-Bus module consists of five functional blocks The Port Address Decoder, Transceiver Set 1, Transceiver Set 2, Control Register and the Status Port. The Capture and Display Module consists of four blocks : The ADC, the DAC, the Analog Switch and the Transceiver Set 3. The System Controller is comprised of three functional blocks The Address Counter, the Dot Clock Generator and the Memory Access Controller

The next four sections are addressed to the implementation aspects of each of these modules. Section 4 5 presents some guidelines for the design of the PCB and the placement of chips The actual PCB layout is given in Appendix A 1 The concluding Section 4 6 narrates the design of the device driver necessary to make the card functional A 'C' language listing of the device driver is given in Appendix A.2

4.1 PC BUS INTERFACE

The implementation of the blocks constituting this module is shown in Figure 4.1

4.1.1 ***Port Address Decoder*** : This functional block receives signals from the address lines, IOW, IOR and AEN lines and decodes these to generate the control signals PA1, PA2, PA3 and PA4 which are required for writing into CR, reading from SP and enabling TS1 and TS2 with proper directions (through IORB) as indicated in the previous chapter.

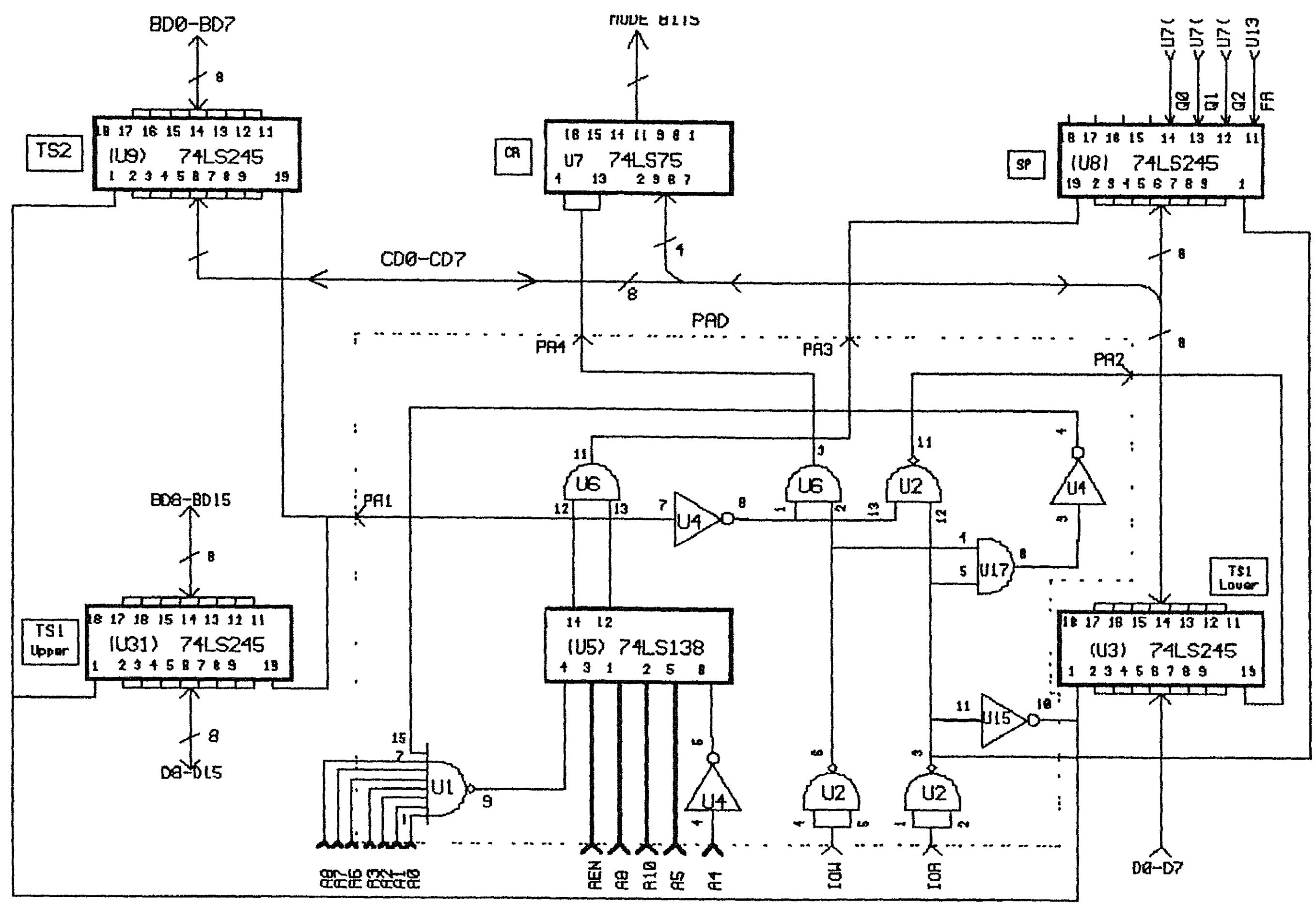

Figure 4.1 PC-BUS INTERFACE MODULE

Before proceeding further it is necessary to decide upon the addresses for these ports Such port addresses can be chosen in any of the reserved space of the IBM PC-AT Referring to the recommended space for I/O port interface given in PC-AT Technical Reference [6], the port address for the CR/SP can be chosen as 7CF (hex) and that for the TS2 can be chosen as 3CF (hex) The portion of TS1 on the lower lower half of the PC data bus should be enabled (with proper direction) for both of these port addresses However, the portion of TS1 on the upper half of the data bus should be enabled (with proper direction) only when TS2 is enabled Full decoding of the address bus is useless as the standard cards on the IBM PC themselves do not support these Thus no additional card can be designed for anywhere outside the recommended space as this will give shadows with the existing cards

The Port Address Decoder has been implemented using a 74LS138 chip (U5) and a combination of few gates (U1,U2,U4,U6,U15,U17)

4.1.2 ***Transceiver Set 1*** This can be implemented using any set of transceivers fast enough to reproduce the data within the duration of the IOW or IOR pulse The actual implementation uses two 74LS245 chips (U3 and U31)

4.1.3 ***Control Register*** · The control register receives the mode selection code from the PC and stores that till it is changed The output bits of this register are decoded by the MAC to control the operation of various other units on the card As only three control bits are required for selecting either of these five modes (see the Table 3.1) a four-bit latch is sufficient to

act as the control register A 74LS75 quad bistable latch (U7) is used for this purpose

4.1.4 ***Status Port*** The function of this block is just to pass on the mode bits of CR and the frame acquired (FA) bit from the MAC to signal the status of the frame capture (once a FC command is given) to the PC as and when required As both of these informations are already coming from latches or flip/flops there is no need to use a latch for this purpose. Although only four bits are needed, in order to keep room for adding more status bits in the future an 8 bit transceiver chip 74LS245 (U8) is used for its implementation.

4.1.5 ***Transceiver Set 2*** The purpose of this Transceiver is to allow the PC to access CR or SP through the Control/Status Bus while the Buffered Data Bus is busy transforming data in the FC and FD modes It should also be fast enough to reproduce the data within the duration of the IOW or IOR pulse in the FO and FI modes The actual implementation is through a 74LS245 chip (U9)

4.2 FRAME MEMORY The Frame Memory stores the frame coming from the camera in the mode FC and reproduces the frame for display in the mode FD It also sends the stored frame to the PC in the mode FO and stores the processed frame (or a frame *generated* by the PC itself) in the mode FI The address for this memory is generated by the ACN under the control of the MAC.

For the reasons mentioned in Section 3 4 7 the memory should be organised in odd and even banks Thus this means that the LSB ACN0 should be used to select either of these banks Rest of the 15 bits in the counter can be used to generate the address and chip select for a particular chip in the selected bank This suggests that ideally each bank should be implemented through a single chip in order to reduce the physical size of the memory and of the chip select decoding logic, i e a memory chip of capacity 32-K Bytes should be used Such a chip in the static RAM family is quite costly, not readily available and has an access time more than that required here The physical size of the memory need to be small because of the limited area available on the card which is to be fitted in the PC Expansion slot Low cost dynamic RAMs of this capacity l speed are readily available but the need of the refresh logic required with em outweighs these advantages The 8-K byte CMOS static RAM HM-6264 is a good compromise especially for prototype testing As this chip has been packaged in 28 pin DIP, eight such chips required for the 64K Bytes of the frame memory will consume quite a good percentage of the card area, yet the comparatively low cost, low access time (150 ns) and very low power dissipation (50 mW) makes it an ideal choice for the first implementation of the design

The next task is to implement the chip select logic for the organisation of the frame memory based on these chips Each of the memory bank will require four such chips Once a bank has been selected through the LSB, the highermost two bits of ACN are decoded to select one out of the four chips of the bank by a dual 2-4 decoder chip (refer Section 4.4)

The organisation of the memory with the the set of eight HM-6264 chips is shown in Figure 4 2. The Figure 4 2(a) shows the even bank and the Figure 4.2(b) shows the odd bank. The counter is incremented at the positive edge of the clock. Therefore, the active high of the 6264 chip has to be connected to the inverted clock going to the counter in order to output or input the data *before the counter is incremented*.

The read/write logic for the memory chip requires that in the modes FC and FO the write line of the memory should be active while in the mode FD and FI the read line of the memory should be active. This can be done for all chips together as the chip select logic enables the particular chip to be read or written

The timing diagram for the chip select generation in the mode FC has been shown in the figure 4 3 (a). For the mode FD this cycle of timing will be repeated till the mode is changed Figure 4 3(b) shows the same for the modes FI and FO.

4.3 CAPTURE AND DISPLAY MODULE

The implementation of this module has been shown in Figure 4 4. A brief explanation of each functional block in this module is given below.

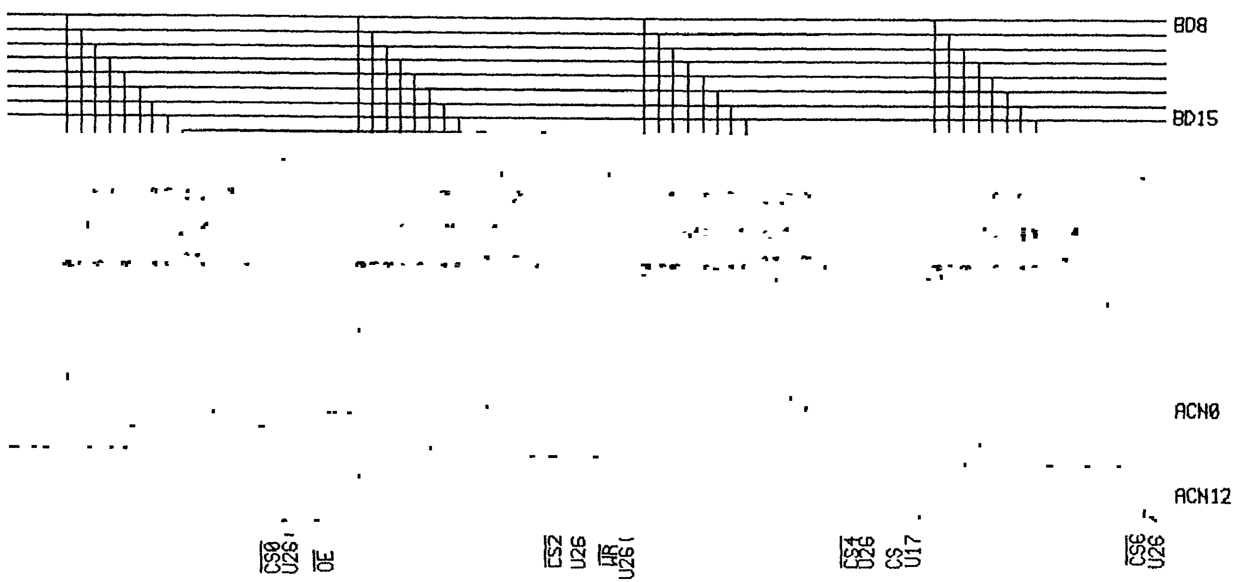

Figure 4.2(a) EVEN MEMORY BANK

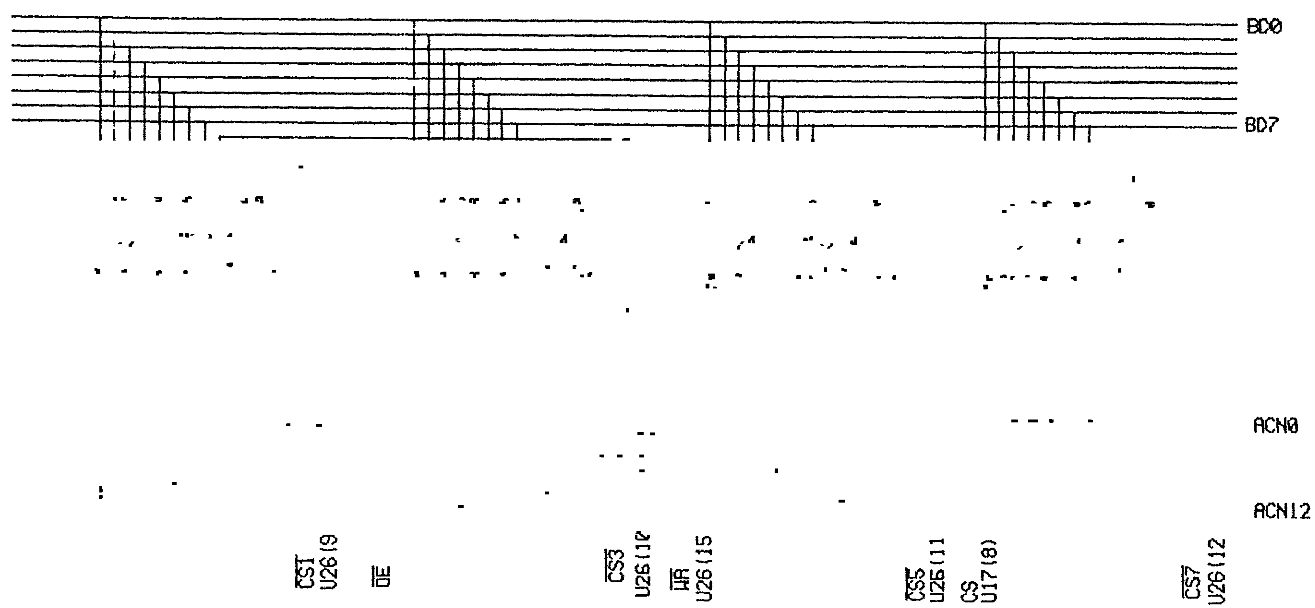

Figure 4.2(b) ODD MEMORY BANK

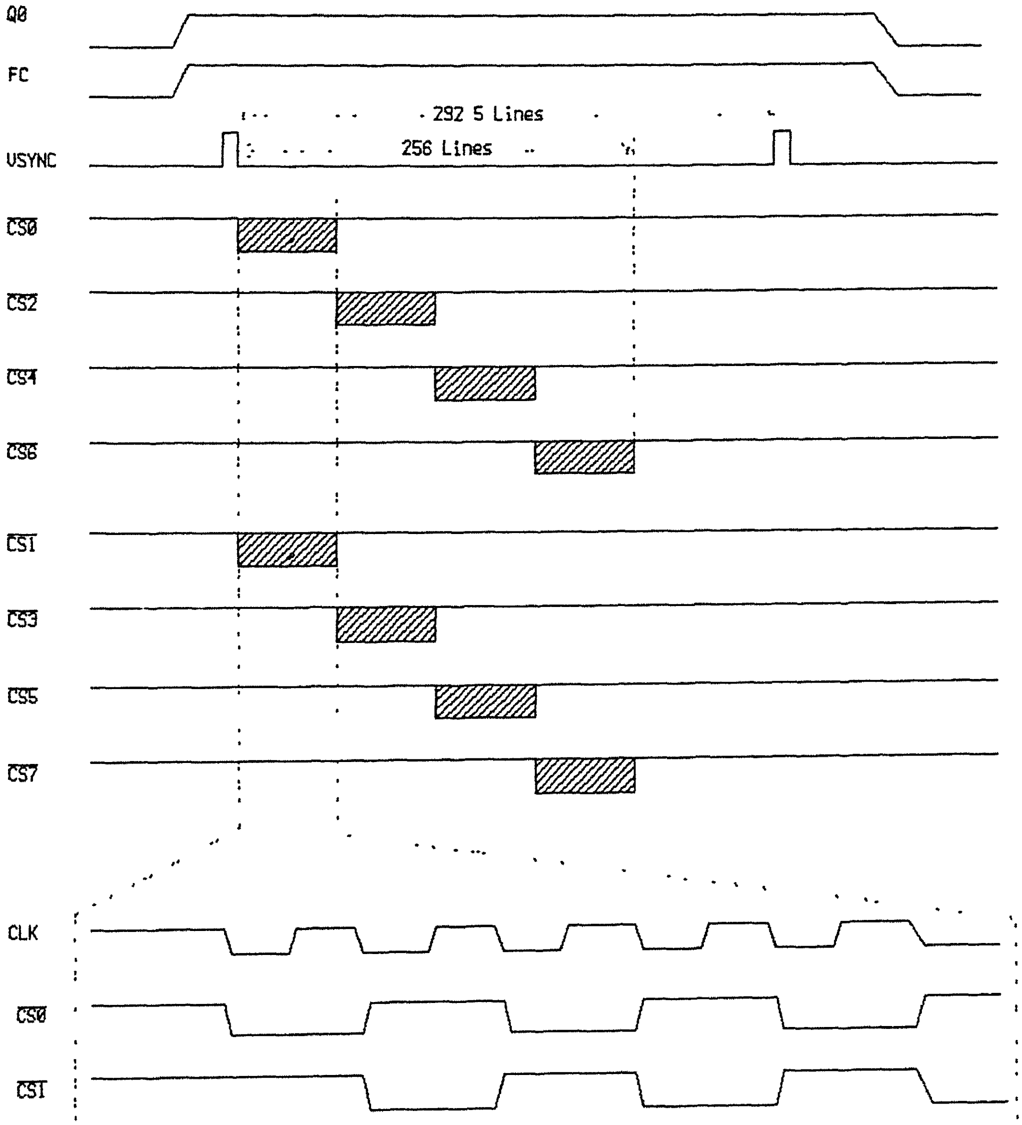

Figure 4.3(a) MEMORY CHIP SELECT TIMING DURING THE MODE FC

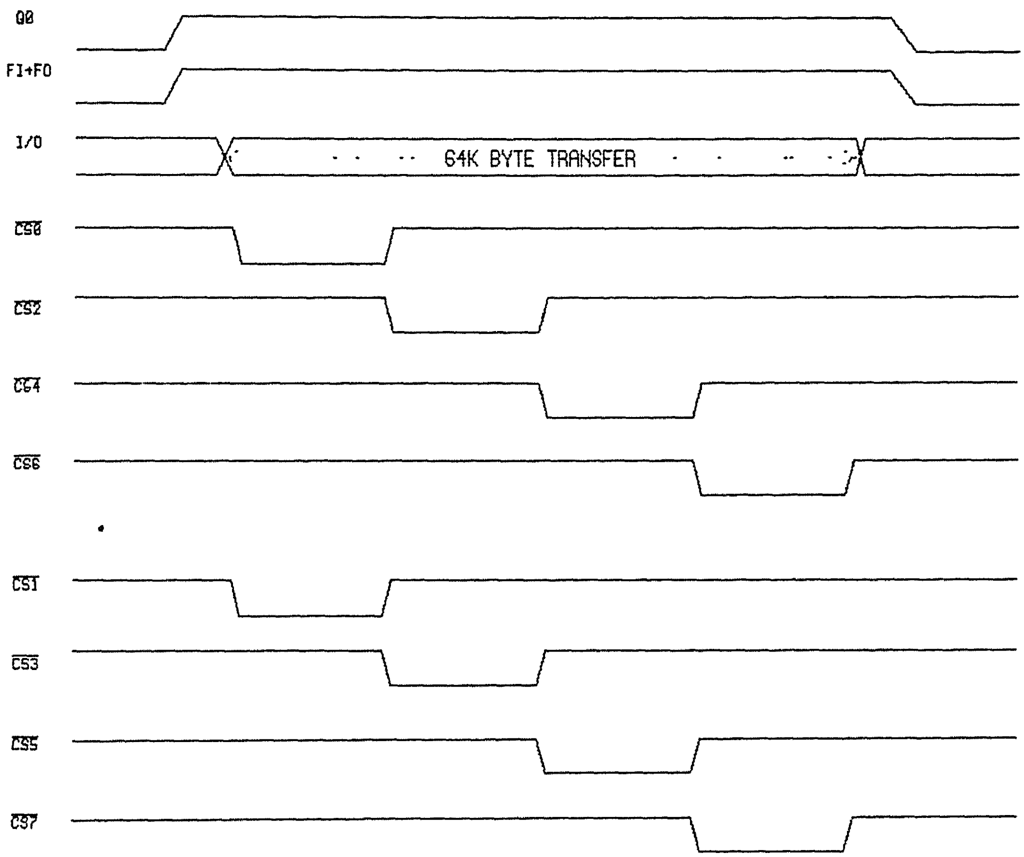

Figure 4.3(b)   MEMORY CHIP SELECT TIMING DURING FI AND FO

4 3 1 ***ADC*** This block which digitises the camera signal and forwards it to the Frame Memory through TS3 can be implemented through a single chip For the 5M samples/second video data only a flash converter can be used Most of the flash converters going upto the digitisation frequency of 5 MHz or more are costly as well as bulky They are not easily available too because of their very low demand The Frame Memory has the capacity to store one byte per sample and therefore an 8-bit converter giving 256 gray levels is needed A 6-bit converter giving 64 gray levels is, however, adequate for most applications (refer Section 2 3) One such low resolution converter is CA 3306 This is a 6 bit flash ADC with maximum digitisation frequency of 15 MHz [6] These features combined with its relatively lower cost makes it an ideal choice for the ADC block

The sampling clock for the ADC comes from the clock selector which forms part of MAC The chip select for the ADC (ADCE) which is active high comes from the System Controller which gives a decoded output for the mode FC CA 3306 has got an onchip output register which drives the digital output lines at the high to low transition of the sampling clock if the phase signal is tied to the ground This arrangement can directly be used for this design as shown in Figure 4 4 This is necessary because the data to be stored should remain steady during the *low period of the dot clock for the memory to store the data before the counter is incremented on next low to high transition of the dot clock*

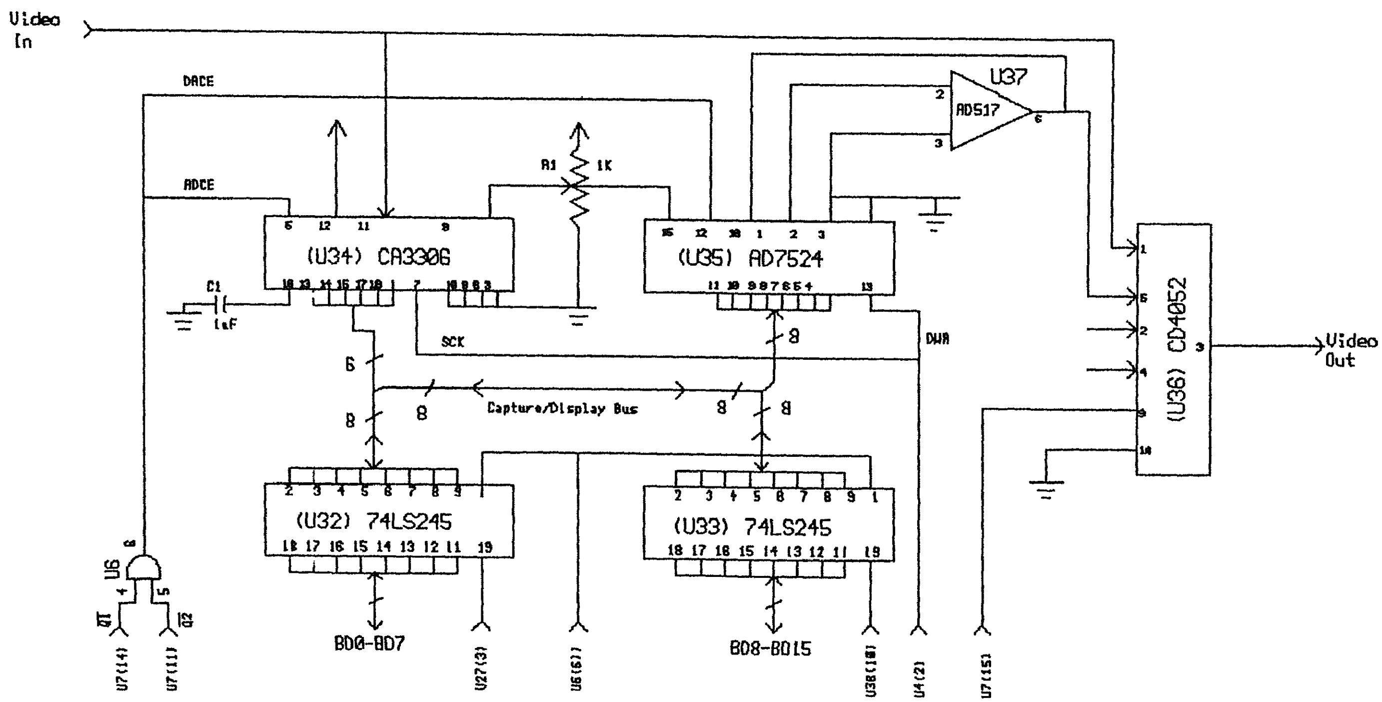

Figure 4.4 CAPTURE AND DISPLAY MODULE

4.3.2 *DAC* This block reproduces the stored picture signal in the mode FD for display by the video monitor. It should also have a settling time of 200 ns (corresponding to a frequency of 5M samples/second) or less It should have onchip data latches and chip select signal also. Finally it should also be cost-effective and readily available. A survey of the chips meeting these requirement suggests AD7524 as a possible choice [7] As it gives its output in the form of a current, an Op-Amp is necessary for converting that to a proportionate voltage Op-Amp AD517 has a slew rate which is compatible with the requirement of the 5 MHz input signal The chip enable for AD7524 is active low and as the modes FC and FD are mutually exclusive

$$ADCE = \overline{DACE} = \bar{Q}_1 . \bar{Q}_2$$

4.3.3 *Analog Switch* : The Analog Switch is simply used to choose the video signal to be sent to the monitor in the different modes. Its channel select lines are simply the mode bits Q_1 and Q_2 resulting in a final video output of the frame grabber as follows :

Mode	Video Output Signal
FC	Camera Output
FD	DAC Output
FI	Zero
FO	Zero

It is important to note that the VSYNC and HSYNC pulses for the monitor are coming from the camera in both of the modes. The commonly available chip

CD4052 is well suited for this purpose.

4.3.4 ***Transceiver Set 3*** This necessity of this set of transceivers has been explained in Section 3 4 9 This set can be implemented through commonly available transceiver chip 74LS245 (U32 and U33) as shown in figure The chip enable for the set connected to the even bank will come through the LSB of the ACN and for the one connected to the odd bank will come through complement of the LSB The direction for each can be set through any of the decoded bit designating the mode FC or FD depending upon the orientation of the chip

4.4 SYSTEM CONTROLLER

4.4.1 ***Address Counter :*** This counter is used to address a 'dot' on the screen As mentioned in Section 2.3, the present design is for a video frame consisting of 256 lines comprising 256 dot each The Address Counter therefore consists of an 8-bit Dot Counter (DCN) and an 8-bit Line Counter (LCN) in tandem However, the two counters have to be properly synchronised with the HSYNC and VSYNC pulses coming from the camera in the FC and FD modes. This is achieved by controlling the Clock, Enable and Reset inputs of DCN and LCN separately through appropriate logic. The Clock is also qualified through a clock selector logic (which is a part of the MAC) in order to choose the source of the clock for the various modes As shown in Figure 4.5 the Address Counter is implemented through four 74LS191 4-bit counter chips

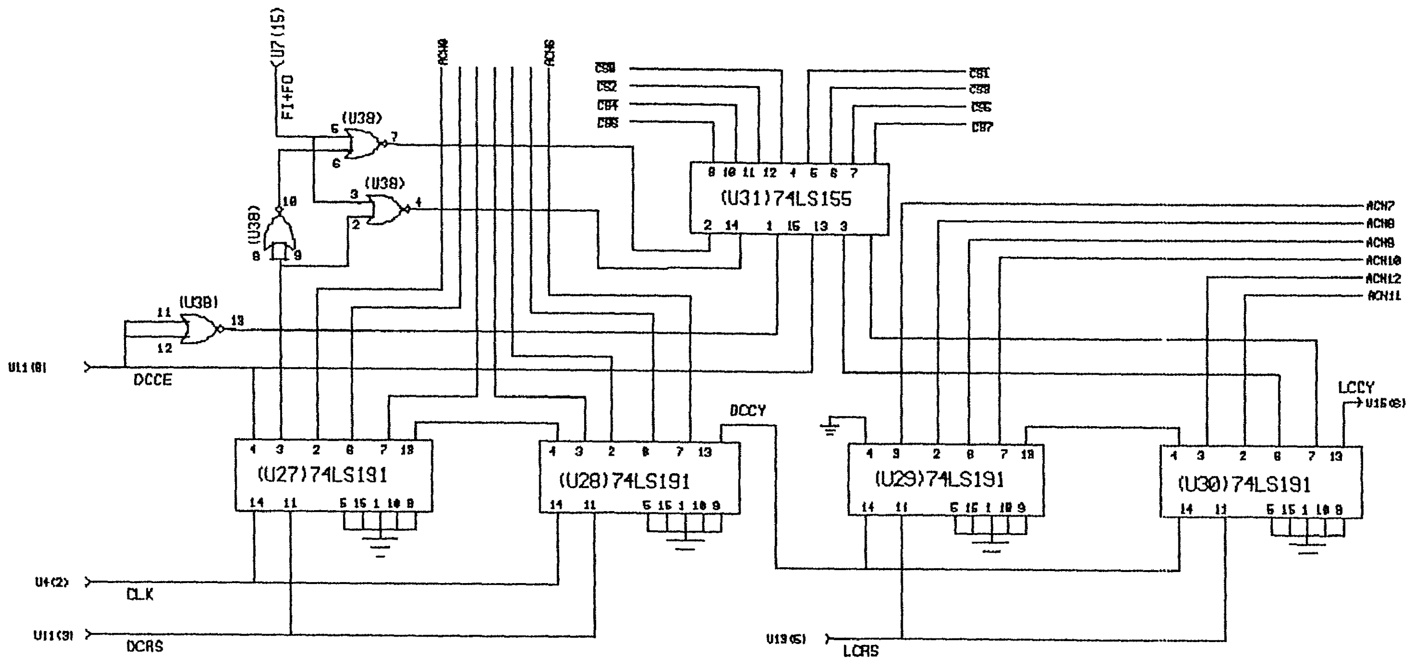

Figure 4.5 COUNTER AND MEMORY CHIP SELECT LOGIC

(U27,U28,U29,U30) Few output lines of the Address Counter are used for generating chip select signals for the eight HM-6264 memory chips (refer Section 4 2). This is implemented through a 74LS155 dual 2-4 Decoder Chip along with minor gating circuitry (U26 and U38).

4.4.2 ***Dot Clock Generator*** : In the present implementation the 5 MHz dot clock is being generated from outside the card This is helpful to test the prototype (especially the MAC) at other dot clock frequencies also However, a provision has been made for a single chip clock generator onboard

4.4.3 ***Memory Access Controller*** : This module consist of three sub-units viz the Synchronisation Logic, the Clock Selector and the ADC/DAC Enable

Synchronisation Logic . This sub-unit is used to synchronise the enable and reset signals for DCN and LCN in the modes FC and FD In the mode FC it sets the bit FA once a frame is captured in the memory.

Inputs to this sub-unit are

(i) VSYNC pulse

(ii) HSYNC pulse

(iii) Mode Control Bits Q_0, Q_1, Q_2

(iv) The Carry from DCN (DCCY)

(v) The Carry from LCN (LCCY)

While VSYNC is active low, HSYNC is active high Both DCCY and LCCY

are active high

The outputs from this sub-unit are

(i) The Chip Enable of DCN (DCCE)

(ii) The Reset input of the DCN (DCRS)

(iii) The Reset input of the LCN (LCRS)

(iv) The Frame Acquired bit (FA)

Except FA which is active high, all other outputs are active low

The implementation of this sub-unit has been shown in Figure 4 6. As the operation of this circuit is quite involved, a state diagram has been given in Figure 4 7 to show the cycle of operation of the circuit in the modes FC and FD It is important to note that in the modes FI and FO this circuit is inactive i e. the DCN is always enabled and the reset inputs to the DCN and LCN is always high during these modes (the FA bit plays no role in these two modes)

Each state in the state diagram represents the set of outputs of the five D-flip flops (U10,U13 and U16) The circuit has only six states and thus the same operation can also be implemented through a set of three D-flip flops However, as this brings no substantial saving in the overall implementation, a set of five D-flip flops are used in order to simplify the design The outputs of these flip flops are denoted by P, R, S, W and X. It is important to note that in the state diagram HSYNC is denoted by 'H' and VSYNC is denoted by 'V'. Furthermore an 'up' or 'down' arrow used as suffix after these symbols designates the rising or falling edge.

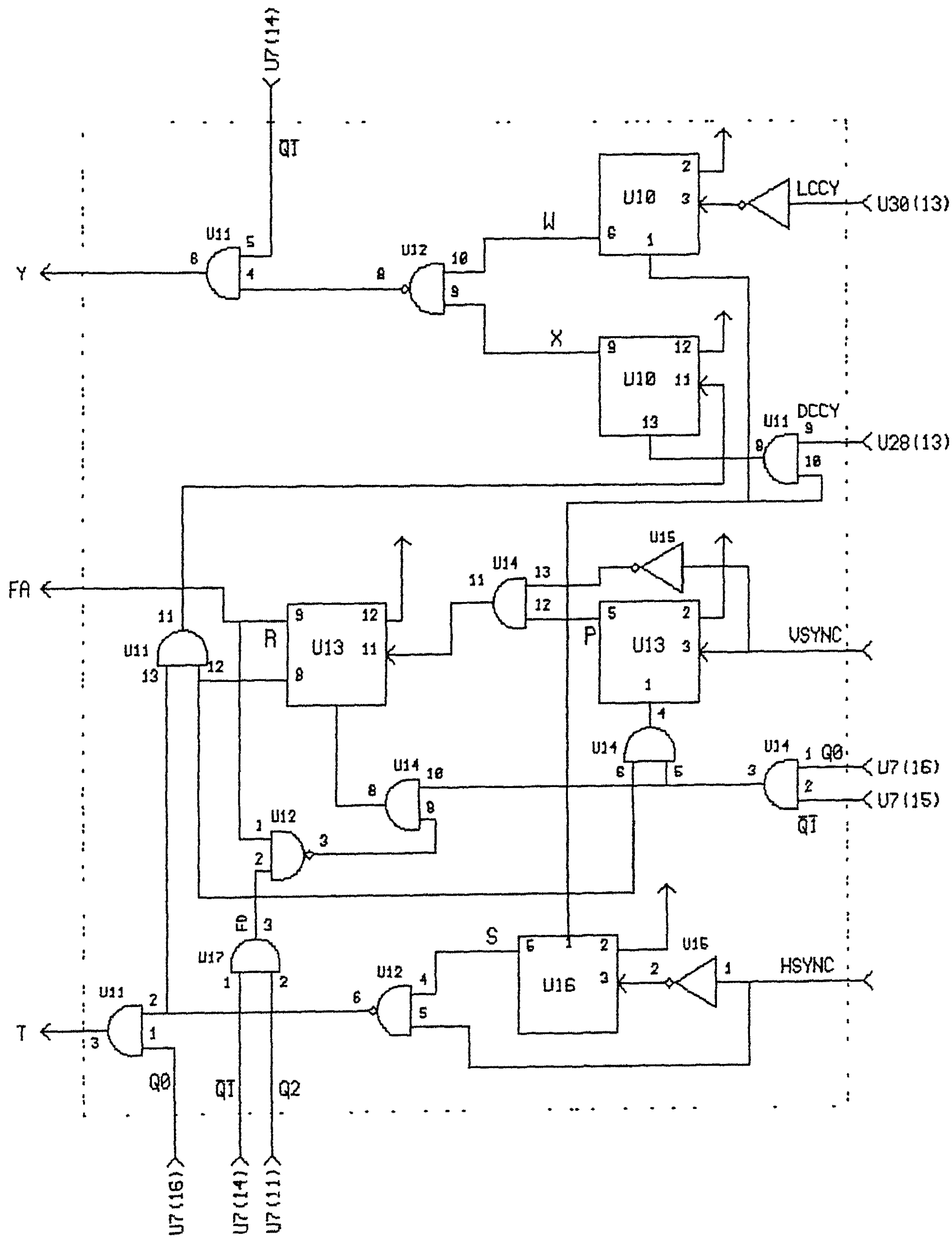

Figure 4.6 SYNCHRONISATION LOGIC

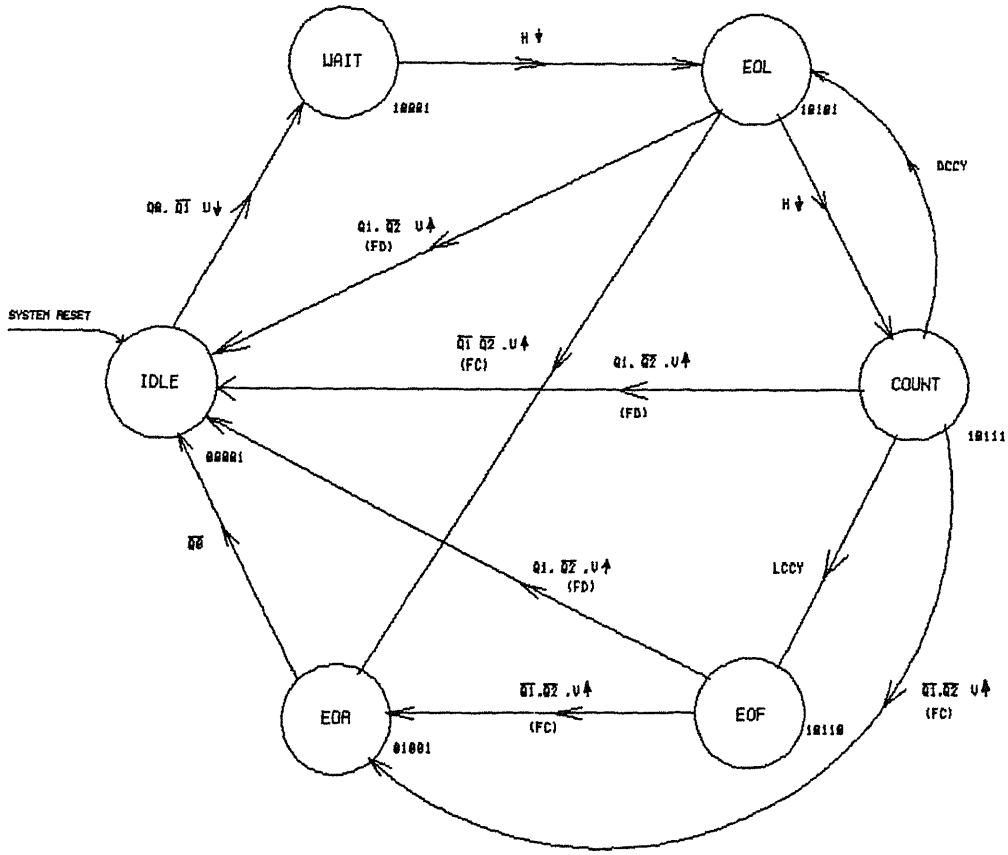

Figure 4.7 STATE DIAGRAM OF THE OPERATION OF THE SYNCHRONISATION LOGIC

Table 4 1 gives a description of each of these six states Table 4 2 tabulates the condition of each output in these six states

TABLE 4.1

STATES OF THE MEMORY ACCESS CONTROLLER

State Name	PRSWX	Description
IDLE	00001	Reset the whole controller
WAIT	10001	Wait for the the Falling Edge of the first HSYNC
EOL	10101	End of a Line (Digitised or displayed)
COUNT	10111	Enable DCN to count dots on the current line
EOF	10110	End of a Frame (Captured or Displayed)
EOA	01001	End of Acquisition (FC only)

TABLE 4.2

STATE V/S OUTPUTS

State	Outputs				
Name	PRSWX	DCCE	DCRS	LCRS	FA
IDLE	00001	1	0	0	0
WAIT	10001	1	1	1	0
EOL	10101	1	HSYNC	1	0
COUNT	10111	0	HSYNC	1	0
EOF	10110	1	HSYNC	1	0
EOA	01001	1	1	0	1

Clock Selector The clock selector receives as the the input the mode information and reproduces as the output the appropriate clock for the ACN (out of IOW, IOR and dot clock) Thus it need only consist of a gating circuit in order to carry out this decision. However care should be taken so as not to propagate any of the clocks through too many gate delays as this will affect the synchronisation The implemented logic has been shown in Figure 4 8.

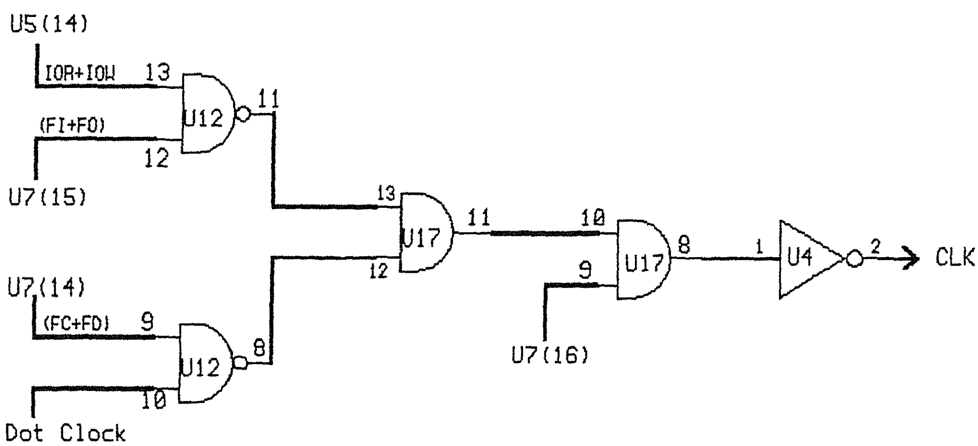

Figure 4.8 CLOCK SELECTOR FOR THE ADDRESS COUNTER

ADC/DAC Enable . This is implemented through a simple decoded gate as has been shown in Figure 4 4 As input this unit obtains the mode information from CR and controls the signals ADCE and DACE ADCE is active high and is connected to the output of a gate (U6) which decodes the mode control bits Q_1 and Q_2 to generate a high signal during the mode FC DACE is active low and thus can be directly connected to the ADCE At this point it is important to note that the chip enable for the transceiver in TS3, which is connected to the even memory bank comes from the counter bit ACN_0 while for that connected to the odd memory bank comes from the complement of ACN_0. The direction control of both of these chips are connected to the ADCE which is high during the mode FC and low otherwise.

4.5 GUIDELINES FOR THE PCB DESIGN

The above implementation of the frame grabber grabber design has been done keeping in mind that the PCB is to be fitted in the PC-AT slot The 36 pin AT extension bus slot is needed only to tap at the upper half of the data bus All other signals to and from the PC are to be tapped from the 68 pin XT slot Thus using this PCB on the XT (with slight modification in the device driver) will give access to only half of the memory (Odd bank) and thus the image will have a resolution of 128x256 Rest of the operation of the PCB will remain the same except with the reduction in the speed of I/O

The complete implementation of the design requires 38 chips (8 out of which are 28-pin each). Such a density of chips on the PCB will require a

multi-layer design However, the prototype testing can be done using a two-layer design of the PCB keeping the other layers in jumpers This requires extreme care during testing and general handling as it brings down the reliability of the PCB Such a two-layer implementation of the design and the placement of various chips on the PCB are shown in Appendix A 1. It is important to keep the Port Address Decoder and the Transceiver Set 1 as close to the appropriate connectors as possible

4.6 DEVICE DRIVER

The device driver can be written in either assembly or 'C' language Writing in 'C' is a better option from the point of view of ease in interfacing the driver with the high level data structures and functions required for preprocessing, feature extraction and feature interpretation

The PCB can be put in a certain mode by sending the appropriate control byte (as given in Table 3 1) to CR at the port address 07CF(hex) The status of the PCB can be obtained by reading the status byte from SP at the port address 07CF(hex) The MSB of the status byte is the bit FA while the next three bits are the Q_2 Q_1 and Q_0 bits out of the Control Register The other four bits are don't cares. *It is very important to put the PCB in mode ID while switching it from one mode to the other*

For the mode FC, once the code is written in CR, the program can wait for an approximate period of 20ms and then poll the MSB of the status byte

(bit FA) till it is high For the mode FD, the PCB will display the frame autonomously till an ID mode command is received

For the Mode FI, once the code is written in CR, the program can transfer the frame to the PC memory through repeated <u>word</u> read from the port 03CF(hex) 64K times For the mode FO, once the code is written in CR, the program can transfer the frame from the PC memory through repeated <u>word</u> write to the port 03CF(hex) 64K times

CHAPTER 5

MOTIVATION & BRIEF LITERATURE SURVEY

This chapter briefly surveys 3-D object recognition techniques in the context of robot vision, discusses their limitations and motivates the technique proposed here the details of which are given in chapter 6

5.1 3-D OBJECT RECOGNITION

3-D object recognition is a rather nebulous term A brief survey of the literature on this subject demonstrates this point [9] Some schemes handle only single presegmented objects, whereas others can interpret multiple object scenes. However, some of these schemes are really performing 2-D processing using 3-D information There are systems which require objects to be placed on a turn-table during the recognition process A few published methods even

require that intermediate data be provided by the person who is operating the system Some techniques assume that idealized data will be available from the sensor and intermediate processors Others require high contrast or backlit scenes Most efforts have been limited recognize polyhedra, spheres, cylinders, cones, generalised cones [14] or a combination of these Many papers fail to mention how well the proposed method work when the scene has a large number of objects (e g atleast 20 objects) Therefore a rigorous mathematical formulation of the problem is very necessary

5 1.1 *Mathematical Problem Formulation*

It is often helpful to define a problem in a stricter mathematical form to eliminate possible problem ambiguities [9]

First we approximate the world for the robot as consisting of N_{obj} *distinguishable* objects We refer to the ith distinguishable object as $\mathbf{A_i}$. We define the origin of the object coordinate system at the center of mass of the object with the three orthogonal axes aligned with the principal axes of the object because these parameters can be precisely determined for any given solid object or solid object model

Also there is a World Coordinate System that is placed at any convenient location Objects are positioned in space relative to this coordinate system by means of translation and rotation parameters We refer to the translation parameters of an object as the vector $\vec{\alpha}$ and to the rotation parameters of an object as the vector $\vec{\theta}$. For the 3-D object recognition we write the necessary six parameters as follows

$$\vec{\alpha} = (\alpha, \beta, \gamma) \text{ and } \vec{\theta} = (\theta, \phi, \psi)$$

We define our world model W as a set of ordered triples (object, translation, rotation) :

$$W = \{ (A_i , \vec{\alpha}_i, \vec{\theta}_i) \}_{i=0}^{N_{obj}}$$

If a time- varying model is required, all objects and their parameters can be functions of time Also the sensor itself is considered an object This is the object A_0 with position $\vec{\alpha}_0$ and orientation $\vec{\theta}_0$. We denote the set of all objects, the *object list as* $L = \{ A_i \}$. The set of all translations is denoted as R^t, and the set of all rotations is denoted as R^r. In the 3-D object recognition problem, $t = 3$ and $r = 3$ R is the set of all real numbers.

With both the depth sensor (or range finder) and the intensity image sensor (a video camera for instance) the scene undergoes a projection transformation. We model this projection as a mathematical operator P, which maps elements in the set $\omega = L \times R^t \times R^r$ into elements in the set of all scalar functions of t-1 variables, which we denote by F: $P : \omega \rightarrow F$
This projection operator might be *orthographic* or *perspective* [10]. As this projection basically transforms the depth information of the scene, these sets F of real - valued functions are referred to as depth map functions, We write the projection as :

$$f(\vec{x}) = P (A, \vec{\alpha}, \vec{\theta})$$

where $\vec{x}$ is a vector of t-1 variables of the focal plane of the sensor. The

spatial parameters of the sensor object (the location $\vec{a}_0$ and the orientation $\vec{\theta}_0$) are implicitly assumed arguments of the projection operator. Two of the rotation parameters (those associated along the axes parallel to the focal plane of the sensor) have a particularly profound effect on this family of functions The 3-D shape of the depth map function changes as the object rotates Translation parameters have no effect under the orthographic projection, and they have negligible effect under the perspective projection unless the sensor is very close to the object of interest (e g. closer than 10 times the maximum object width)

The depth map object recognition problem can now be stated in terms of the modeled world as follows .

Given the world model W with N_{obj} objects and given any realizable depth map functions $f(\vec{x})$, compute the projection set mapping $P^{-1}(f(\vec{x}))$ to obtain all possible explanations of the function $f(\vec{x})$ in terms of the world model

In many cases, there is only one valid scene interpretation. Nonetheless a general purpose vision system must generate a list of all valid scene interpretations whenever ambiguous single view situations are encountered The natural solution to this problem lies in determining the next best view from which sensor data should be acquired so as to eliminate ambiguity. Since there is no general theory regarding the computation of this mapping, researchers are free to choose their own best methods

This formalism is now augmented to state the object recognition problem

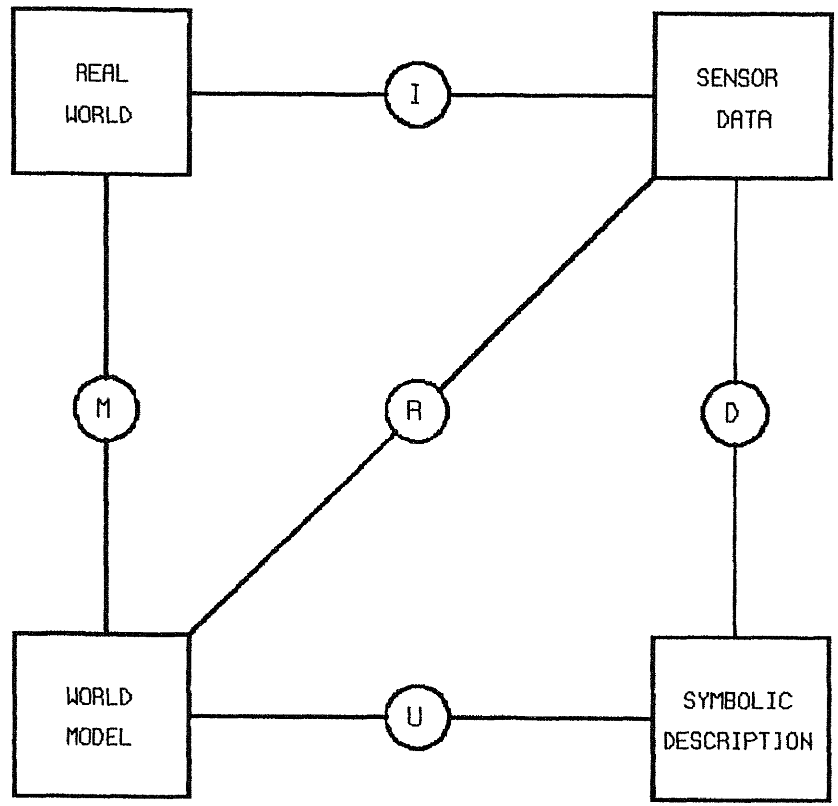

Figure 5.1 COMPONENTS OF AN OBJECT RECOGNITION SYSTEM

assumption about real world object shapes should be incorporated The modeling process (M) provides object models for real world objects. Object reconstruction from sensor data is one method of building models automatically. Neural network [12] based architectures hold promise in this area. Also this method is preferred for a fully automated system but is not required for our problem definition Hence, object models could also be constructed manually using geometric solid modeling programs. The understanding, or recognition process (U) involves an algorithm to perform matching between these models and data descriptions. This process might include interacting data- and model- driven subprocesses, where segmented sensor data regions seek explanations in terms of models and hypothesized models seek verification from the data The rendering process (R) produces synthetic sensor data from object models. A vision system should be able to communicate its understanding of a scene *verbally* and *visually* to any person querying the system Rendering also provides an important feedback link because it allows an autonomous system to check on its own understanding of the sensor data by comparing synthetic images to the sensed images

Any object recognition system can be discussed within the framework of this system model.

5.1.2 *Characteristics of an Ideal System*

An ideal vision system should have the following capabilities [9].

1. It should be able to handle sensor data from arbitrary viewpoint

without giving preference to horizontal, vertical, or other directions this requires a view independent modeling scheme that is compatible with the recognition processing requirements

2 The system must handle arbitrarily complicated real - world objects without giving preference to either curved or planar objects

3 The system must handle arbitrary combinations of a relatively large number of objects in arbitrary orientations and locations without being sensitive to superfluous occlusions (i e if occlusion does not affect human understanding of a scene, then it should not affect the ideal automated object recognition system either)

4 The system must be able to handle a certain amount of noise in the sensor data without a significant degradation in system performance

5 The system must be able to analyze scenes quickly and correctly

6 It should not be difficult for the system to modify the world model data in order to handle new objects and new situations

7 It is desirable for the system to be able to express its confidence in its own understanding of the sensor data

As was mentioned earlier in Section 2 1 no existing vision system has all these capabilities, nor a cost effective system in the near future will posses all these capabilities

The technique proposed in this thesis will be compared with each of these, in order to measure its effectiveness, in the next chapter once it is

introduced

5.2 A SURVEY OF SOME EXISTING TECHNIQUES

Some existing techniques are now surveyed within the framework established in the previous section

Following subject areas relevant to the 3-D object recognition problem are being considered .

- 3-D object representation schemes
- Feature extraction
- Feature interpretation or Object Recognition

This survey is basically meant to introduce these ideas. Details can be obtained from the references cited.

5.2.1 *3-D Object Representation Schemes*

Both Computer Graphics as well as Computer Vision need the same basic type of geometric object information, but the storage, utilization, and level of detail of the information is quite different [9]. Some popular schemes of representation in graphics are Wire - Frame, Constructive Solid Geometry, and Octrees. Computer Vision uses either these or other special techniques like Generalised Cylinder or Sweep Representation, Multiple 2-D Projection

Representation etc

Wire Frame Representation A wire frame representation of a 3-D object consists of a 3-D vertex point list and an edge list of vertex pairs, or it can be formatted as such This representation was quite popular owing to its simplicity. Although fast wireframe displays are still popular, solid modeling has replaced wireframe modeling because wire-frame modeling is an ambiguous representation for determining such quantities as the surface area and volume of an object As shown in Figure 5 2, wireframe models can sometimes be interpreted as different orientations of the same object

Constructive Solid Geometry Representation : Constructive Solid Geometry (CSG) representation of an object is specified in terms of a set of 3-D volumetric primitives (blocks, cylinders, cones and spheres are typical examples) and a set of operations · union, intersection and difference See Figure 5 3 for an example of a CSG description of an object The storage data structure is a binary tree, where the terminal nodes are instances of primitives and branching nodes represent set operations . and positioning information CSG tree define object volume and surface area unambiguously and are capable of representing complex shapes with a very small amount of data. However the boundary evaluation algorithms required to obtain usable surface information are very computationally expensive. Also general sculptered surfaces are not easily represented using CSG solid modelers. A general purpose modeling system must be able to represent such surfaces.

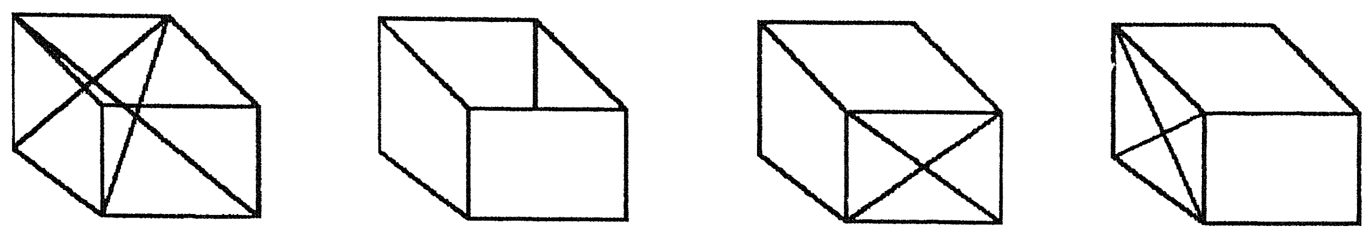

Figure 5.2 DIFFERENT INTERPRETATIONS OF THE SAME WIRE FRAME MODEL

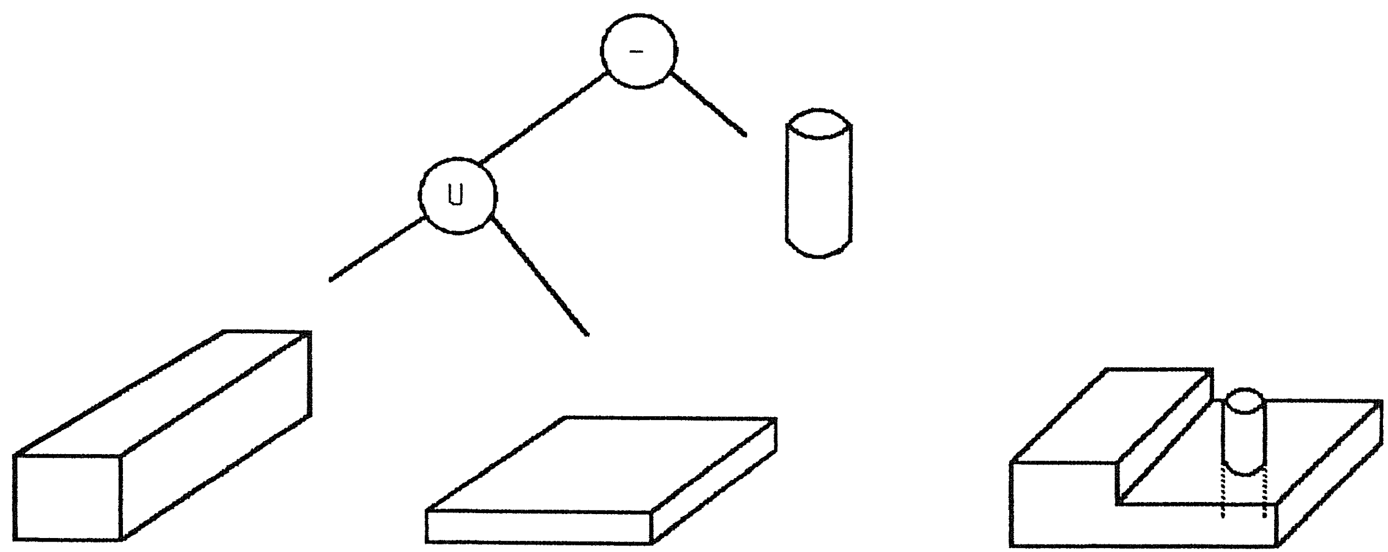

Figure 5.3 SOLID MODEL REPRESENTATION OF COMPLEX SHAPES

Octree Representation [13] . An octree is a hierarchical representation of spatial occupancy Volumes are decomposed into cubes of different sizes, where the cube size depends upon the distance from the root node Each branching node of the tree represents a cube and points to eight other nodes that describe object volume occupancy in the corresponding octant subcubes of the branching node cube The basic idea of octrees has been derived from 2-D trees (usually referred to as quadtree).

Generalised Cylinder or Sweep Representation [14] . Generalised cylinders or generalised cones are often called "sweep representations " because object shape is represented by a 3-D space curve that acts as the spine or axis of the cone, a 2-D crossectional figure, and a sweeping rule that defines how the crossection is to be swept and modified along the space curve. Generalised cylinders are well suited to many real world shapes However, it is almost impossible to represent certain objects as generalised cylinders , for example the body of an automobile or a complex assembly of machine parts Therefore this scheme is not general purpose.

Multiple 2-D Projection Representation : For some applications it is convenient to store a library of 2-D silhouette projection to represent 3-D objects with a small number of stable orientations on a flat light table, this representation is ideal if silhouette of the objects are different enough It is not a general purpose technique, however, because it is possible for many different 3-D object shapes to posses the same set of silhouette projections

A more detailed technique is the ***Characteristic Views*** technique described in [15]. All of the infinite 2-D projection views of an object are grouped into a finite number of topologically equivalent classes and are related via linear transformations. This is a general purpose representation because it specifies the 3-D structure of an object However, it can require a vary large amount of data storage for complex objects. Figure 5 4 shows seven representative characteristic views for a spanner.

Any representation chosen for object recognition should also be compatible with matching algorithms. To be suitable for matching, each real - world object shape requires a unique description within the framework of a given representation Most representations do not guarantee unique numerical descriptions of object shapes

A general overview of the various representation schemes in Computer Vision is given in [16]

5.2.2 *Feature Extraction*

The type of features to be extracted from a preprocessed scene depend upon the gray levels of the image and accuracy of results desired. While features form a simple binary (two gray level) images may be as simple as area, perimeter, moments, Euler number etc. those extracted from shaded images range from lines, circles, ellipses, curves etc. to the more sophisticated but computationally intensive features like texture gradient, intensity gradient, surface shape and extended gaussian image (EGI) [11]. Also

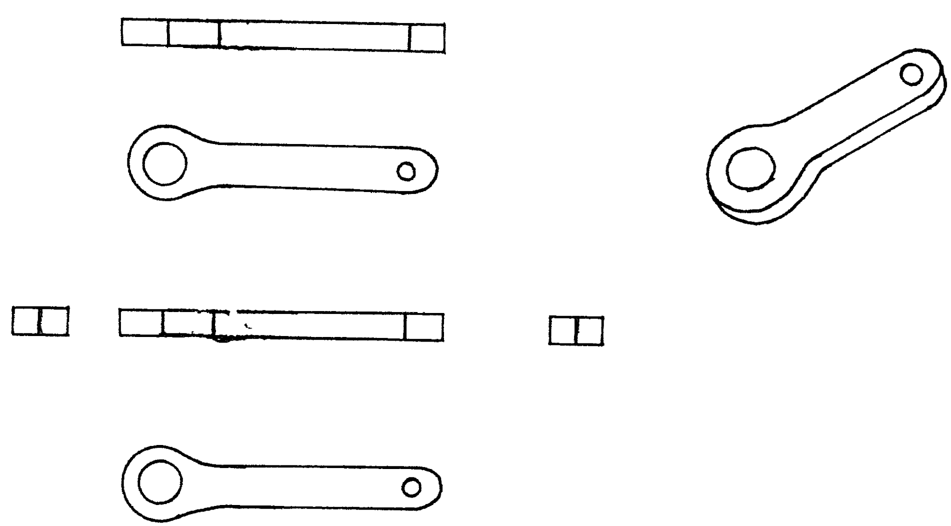

Figure 5.4 SEVEN CHARACTERISTIC VIEWS OF A SPANNER

the amount of preprocessing required depends upon the type of feature to be extracted. Sometimes in order to detect more than one feature a scene may have to be preprocessed through various schemes to detect each feature The features extracted from binary images have severe limitations as these are *rotationally variant*. Though features such as lines, circles, ellipses etc. also suffer from these limitations, the relations between them are free from such limitations It is thus very important to store the rotationally invariant relations among the various features as part of the object model description. Binary image features are quite simple and are explained in various popular texts on Computer Vision However, the features for shaded images as mentioned above are a bit harder and take more computation time Genralised Hough transform [17] is an elegant method for detecting features like lines, circles, ellipses etc But this method is heavily computationally expensive for arbitrary curves Chain encoding, signatures and slope density function [18] are the alternatives for detecting such features

5.2.3 *Feature Interpretation*

Various kinds of feature interpretation techniques have been given in the literature [9]. Many are specialised expert systems The basic approach relies on either a *decision tree* or *pattern classification*.

Decision tree approach : A decision tree approach is based on sequential searching The searching begins from the root node of the tree with some

characteristic feature(s). Depending upon the measurement of this feature a branching is made to the next node. Here some other feature measurement is taken and again a branching to the next node lower in hierarchy is made. The decision based on each feature measurement thus propagates down to a leaf node which represents an object. This object is chosen as the one viewed in the scene In the decision tree, heuristics are also used to branch from one node to the other This often results in faster convergence to the leaf node

Pattern Classification Approach : The pattern classification approach assigns the viewed object to one of a number of known classes once the pertinent features have been measured The basic paradigm is to obtain *n* feature measurements on the entity to be classified and then consider the result as a point in an *n* dimensional *feature space* A *feature vector* is formed by adjoining the feature measurements [11]. Various algorithms have been explored for dividing the feature space into compartments that can be used in classification Most popular ones which do not require heavy mathematical computation are Nearest Neighbor Classification and Nearest Centroid Classification [19] A significant problem with these approaches is the acquisition of enough information to allow intelligent placement of the decision boundaries If underlying probability distributions are available, we can place the boundaries to minimize the error criterion. Usually, though, these distributions are estimated from a finite number of examples drawn from each of the classes, and we can often work directly with this information, rather than estimating probability densities. The underlying assumption here

is that points belonging to the same class tend to cluster and the points belonging to different classes tend to be separated

The strength of the pattern classification approach is that it is a parallel approach. While in a decision tree the decision propagates from the root to the leaf node sequentially, in the pattern classification method the result is reported in one shot without sticking to any sequential searching Thus if each feature measurement can be assigned to one processing unit, the classification unit can immediately combine the measurements taken by each unit to produce the result This kind of implementation for the decision tree is very difficult owing to its inherent sequential nature.

Another attractive feature of the pattern classification method is that it is trainable If the error in the classification is reported to the classifier through manual feedback the classification boundaries can be adjusted through program control The initial phase of this system should be a training period where by showing as many possible views of an object, the system can be trained to form the cluster boundaries by itself

5.3 CONCLUSION

Previous sections discussed 3-D object recognition in general and surveyed some existing techniques.

In robot vision the camera can be mounted on the robot arm itself (or

their may be more than one fixed camera), thus giving more than one view of the same object(s) in the robot workspace There are instances of the moving camera approach with range images [20] No work has been cited in the literature about incorporating moving camera approach in intensity images The range image methods have been quite successful but require costlier and bulkier hardware Using a moving camera with the intensity image as given by the frame grabber *may* thus lead to some fruitful results The image acquired is of 256x256 resolution with 256 gray levels as given by the frame grabber (refer to chapter 2, 3 and 4 for hardware details of the grabber)

Now the questions which a strategy based on this infrastructure should answer are

- Does it help in anyway to identify the objects with better confidence ?

- What features are needed to help identify an object with minimum error if several views of the same object are available and whether such features can be measured without excessive computation or control of the environment ?

- As compared to the already existing methods is it closer to the characteristics of the ideal system laid down in section 5 1 2 ?

The next chapter tries to answer these questions in order to come up with a feasible technique of identification.

CHAPTER 6

TOPOLOGICAL CONTOUR BASED MATCHING TECHNIQUE FOR 3-D OBJECT RECOGNITION

This chapter describes the technique proposed in this thesis. Chapter 5 discussed 3-D object recognition in general, explained the characteristics of an ideal recognition system and surveyed some literature to focus the limitations and positive aspects of some existing techniques Chapter 5 concluded with the fact that the real key to any 3-D object recognition system is the features it looks for and does measurements on This is in agreement with the statement that the results of feature interpretation are only as good as the features selected for measurements No amount of sophistication in the decision algorithm can make up for a poor selection of features [11]

Based on this belief the section 6 1 first discusses the feature selection for 3-D object recognition in general and for the problem addressed

in this work in particular The next section tries to develop the object model representation required for the kind of features that section 6 1 has suggested Section 6 3 proposes the feature interpretation technique a vision system for the features and object model representation suggested in sections 6 1 and 6.2.

6.1 FEATURE SELECTION

For identifying an object a particular type of feature can be ***detectable*** or ***measurable*** . A detectable feature is the one the mere presence or absence of which can immediately identify the object as belonging to a particular class. A measurable feature is one on which measurements have to be made in order to identify the object as belonging to a particular class Thus a *detectable* feature can be considered analogous to a binary variable while a *measurable* feature can be considered analogous to an analog variable. It is possible that a feature may be detectable for one object but not for a second object. For any particular object a detectable feature is said to belong to ***class d*** and a measurable feature is said to belong to ***class m***.

It is always beneficial to find out a class 'd' feature for a particular object. But it may not always be possible to find one given only the intensity image of the object. In such cases several class 'm' features for that object are combined together to form a class 'd feature for the object to help carry out the identification. The real task for 3-D identification of the object is to form such a class 'd' feature with the least complex and minimum number of

class 'm' features available, minimising the amount of computation. The examples given below help demonstrate these concepts

First let us consider the world as consisting of two classes of objects viz polyhedra and symmetrically curved objects The examples of former class are cubes, parallelepipeds and for the latter are spheres, cylinders, cones etc The image of the object is properly preprocessed first to remove noise as far as possible, is passed through high pass filtering and thresholding in order to leave only the contours of the object From such an image a curve is a detectable feature to classify the object as belonging to the latter class No measurement on the length or radius of curvature of the curve is required to infer this. Ofcourse it is important to consider some other feature(s) as well before taking a final decision, since the curve may also be a result of the noise. But its mere presence is a powerful clue for the speedy convergence of the identification algorithm. Going further, once such a curve is detected the presence of a straight line will quickly exclude the possibility of the object being a sphere, and so on. A definite identification can now be made using lines which are measurable features. The presence of two lines may lead the system to infer the object as a cone 'or' a cylinder Thus the identification algorithm has to *make measurements* on the two lines (or has to combine the two class 'm' features to form a class 'd' feature) and has to find out if they are parallel or intersecting, to classify finally the former as belonging to the cylinder and the later as belonging to the cone.

Now let us take complex 3-D objects say a spanner and a ball-bearing.

The preproccessing performed on the image is the same as mentioned in the previous example. There is no easily detectable class 'd' feature that can classify the object viewed as being either of the two or as any third object The contours of both contain lines, circles, ellipses and general curves etc. (see Figure 5 4). Thus to distinguish between the two requires building a class 'd' feature through these class 'm' features A closer observation of the problem suggests that if the object has more than two circles or ellipses the object can not be a spanner Again no detailed measurement or computation with the parameters of these circles or ellipses is required. Similarly if the object has more than three lines the object can not be a spanner and if the object has more than four lines it can not be a bearing also Such measurements on class 'm' features which are available may help the identification algorithm converge to the final decision quite fast. This is in contrast to doing precise measurements through computation on these features to take a decision

All this explanation has been given in order to emphasise the point that the selection of as many class 'd' features as possible (for the particular class of problem addressed) obtained through minimum amount of computation is what should get the maximum attention in 3-D object recognition algorithms. Most of the conventional techniques, however, try to build up such a feature with class 'm' features through extremely complicated mathematical techniques which are computation intensive. In other words a 3-D object recognition algorithm's efficiency can be compared with that of the others by finding out at what stage of the computation does a class 'd' feature emerges At this

point it is important to emphasise that the robot vision problem unlike the general computer vision problem is not 'open-ended'. That is in robot vision we apriori know the set of objects to be present in any scene. Thus it should be always possible (using multiple views in the most cases) to combine some class 'm' features to make up a class 'd' feature.

The task now is to select the features for a particular problem The most easily measurable features to represent 3-D shapes are simple geometrical entities such as lines, circles, ellipses and general curves The latter ones i.e. circles, ellipses and generalised curves are of the class 'd' to distinguish between the two sets of objects viz one having any curved surface in the object and one not having any curved surface. Note, however, that while the presence of a class 'd' feature puts the object in a certain class (e.g curved objects) its absence does not mean that the object does not belong to that class (e g cylinder viewed perpendicular to the axis passing through the ends of the cylinder). Now to identify the particular object in the set selected one again needs class 'd' feature(s) for either associating the viewed object immediately with a model of the stored one or to delete a model from the list of possible objects This can be done either by going for higher level features such as the EGI representation [11], texture etc or by building some class 'd' features on the basis of the class 'm ' features available These approaches usually require highly complex calculations However if the relations among the various geometrical features are generated this can lead to class 'd' features with considerably less computation. The amount of computation depends upon the type of relation chosen which in turn depends

on the representation chosen for the models of objects For example for the CSG representation of the models (refer to section 5.2) the amount of computation to generate the relations is fairly large as compared to that required for the characteristic Views representation specially for objects which are not too complex In terms of the physics of image formation this can be interpreted as the relation chosen should be the one which is rotationally invariant So whether the choice of these features really lead to a better result can only be seen after finding out the best representation for the models The choice of such a modeling scheme is explained in the next section

6.2 THE CHOICE OF THE MODEL REPRSENTATION SCHEME

Section 5.2 describes some commonly used representation schemes for 3-D object recognition systems and discussed their relative merits and demerits This section attempts to pick one of these and modifies it to suit the features selected in Section 6.1

The wire frame scheme can immediately be rejected This scheme is highly ambiguous as it involves too many class 'm' features i.e numerous lines and curves it uses to represent the object A CSG scheme is also not an attractive choice This scheme is suitable for features based on volume elements as it represents the objects in terms of the primitive volumes Thus it is more apt for features such as EGI and texture as these features can be transformed to class 'd' feature for volume elements with minor computation However, the measurements required for these features themselves will

require enormous amount of computation. The octree scheme of representation is more suitable for describing the robot's workspace but not for the object identification by the robot Generalised cylinder scheme has already been rejected in Section 5 2 1 owing to its shortcoming for describing complex shapes. The characteristic view technique (refer Section 5 2), however, seems promising mainly because it represents the objects in terms of the lines, circles and general curves and the *relations among them*. This is because it stores the silhouette of the 3-D view of the object in the form of 3-D information and not in the form of existing 2-D schemes Also the details required to represent this information are few as it only stores the contour information It does not give any consideration to what is inside those contours. In other words it tries to capture that information from a view of the object which is rotationally invariant for a large range of the angle of rotation of the camera (or of the object) If the camera (or the object) is rotated beyond this range the view changes suddenly to another characteristic view which behaves the same way for another range of rotation This scheme has a serious shortcoming, however, that the number of characteristics views of an object grows very fast so that managing with such objects becomes very difficult. So before finally choosing this as the scheme of representation for the features selected it is important to present this in a more logical form. This is necessary in order to incorporate some modifications in this representation scheme , by putting some constraints on the objects (especially the man-made objects as discussed in the Section 2.2) and taking advantage of the moving camera mounted on the robot arm.

Event and ***Visual Potential*** [21] · The above ideas can be presented in a different but more logical form ***Aspect*** is a unique toplogical struture which represents a set of views of the object. For almost any vantage point, small movements do not affect aspect A change in aspect is referred to as an ***event*** An object is described by a graph, referred to as ***Visual Potential,*** where the aspects form the nodes of the graph, and events form the arcs of the graphs. Visual object complexity is measured using the diameter of the visual potential (or aspect graph). Figure 6.1 adopted from [21] shows the aspect graph for a cube ; in this case there are three types of aspects : one, two or three faces are visible

Any modification in this representation scheme should try to reduce the diameter of this graph for a particular object using some outside knowledge and constraints. An example will help clarify this concept before proceeding to make any such modification

Let us first take the example of the cube as mentioned above. For such an object this can be done through the following steps :

1 By making the measurements on the features in a form which is independent of the 2-D coordinate system of the image plane of the camera This will take care of the redundancy in the visual potential which creeps in because the two topologically equivalent views taken from diametrically opposite sides of an object. This will also account for the scaling and rotation transformation brought in the characteristic views

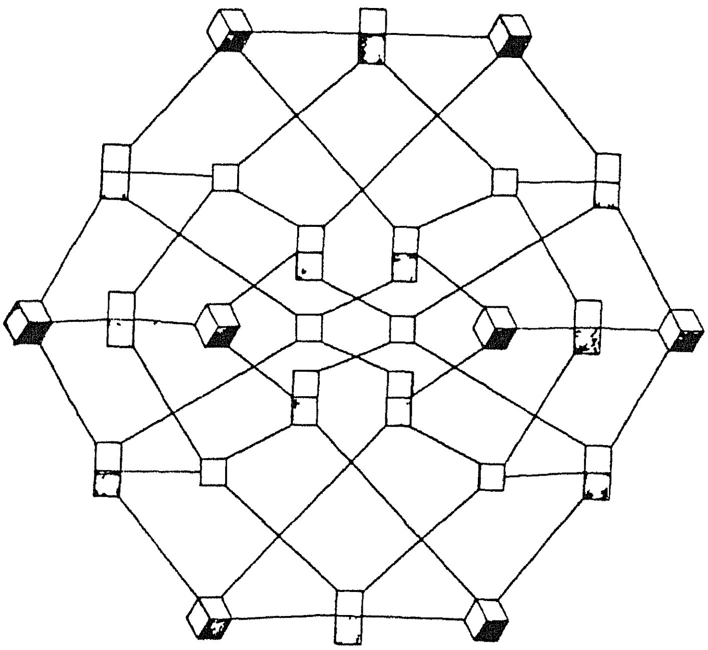

Figure 6.1   VISUAL POTENTIAL OF A CUBE

as a result of the change in the location and orientation of the object.

2 By storing the relations among the various class 'm' features as another set of features Thus in the cube's case as given above a quadrilateral which represent the set of the characteristic views with only one face visible can be represented by a set of four lines and four points of intersection among them A set of 5 lines and 6 points of intersection will represent the set of characteristic views of the cube with two faces visible Finally a set of 9 lines and 7 points of intersection will represent the set of characteristic views with three faces visible

This may lead one to argue that there may be an infinite number of objects which may give rise to this description The situation in robot vision does not suffer much from this limitation A typical robot vision system is supposed to handle at most 50 objects at a time and there is no other object out of these 50 objects that enters the field of view of the robot. If say the cube is one out of these 50 objects the robot vision system need only distinguish it from the other 49 objects In case of conflicts some other features can be looked for and another representation can be made and compared in the same fashion to converge on the decision. Another alternative which is specially attractive to the robot vision system with the camera mounted on the robot arm is to move the camera around the object by a large angle and repeat the above process of checking against any conflict Two or three such views will usually give rise to the desired results.

With this kind of representation for each characteristic view the aspect graph or visual potential can be modified by deleting all but one of the

views which have the same description This means the visual potential now consists of only three nodes or aspects one each for one, two or three faces of the cube visible.

The need now is to generalise this representation scheme for any set of objects The following algorithm attempts to do this

procedure represent()

begin

for each object

begin

Construct the visual potential of the object Represent each node of the visual potential in terms of the no of lines, no. of points of intersections among them, no of circles, no of ellipses and no. of arbitrary curves etc ,

end

for each object's visual potential.

begin

Delete all but one node of the set of nodes having the same representation ,

end

for each object's visual potential

begin

If (the representation of any of its node is the same as that of any of the rest of the visual potentials)

begin

make a link between the two nodes,

end

end

return(),

end

Any set of objects can be represented with this scheme Now the identification algorithm's effort is to find a node which does not has a link to any other node In case there is such a link the viewed object corresponds to any of the two or more number of objects corresponding to these nodes. The best approach to resolve this conflict is to move the camera by a large angle and repeat the process till a linkless node is found. Specially if dealing with occlusion it is always worthwhile to move the camera to two or three locations before taking any decision

The final question before choosing such a representation scheme is whether it is reasonably efficient for representing relatively complex objects like washers, screws, nuts, bolts, bearings, shafts etc ?

For all these primitive objects which form part of a typical assembly advantage can be taken from the fact that *these symmetric in almost all of the diametrically opposite directions of viewing This means the objects having this property will have a visual potential usually consisting of four nodes only These will correspond to the plan, elevation, end elevation and the isometric view of the object* An object which is not symmetric is very difficult to be machined and is usually *assembled* with symmetric primitive objects Figure 6 2(a) (b) and (c) demonstrate this point Thus as far as these kinds of primitive objects are concerned this problem is solved Infact a simple subassembly assembly made of such components will also obey this property The situation, however, changes for bigger assemblies Even in such situations the improvement gained by using the above modifications in the characteristic view over the conventional method reduces the diameter of the visual potential of such an asymmetric object by almost an order of magnitude Further work is required (i) to generate the views which form the node of the modified visual potential through some automatic means using those of the primitive components in the assembly and (ii) to reduce the diameter of the visual potential even further by putting constraints and using knowledge about the scene For example one such constraint can be put by knowing that the assembly can appear only in one or two stable positions. This is true for most assemblies which have to be positioned with their base on the ground. Thus

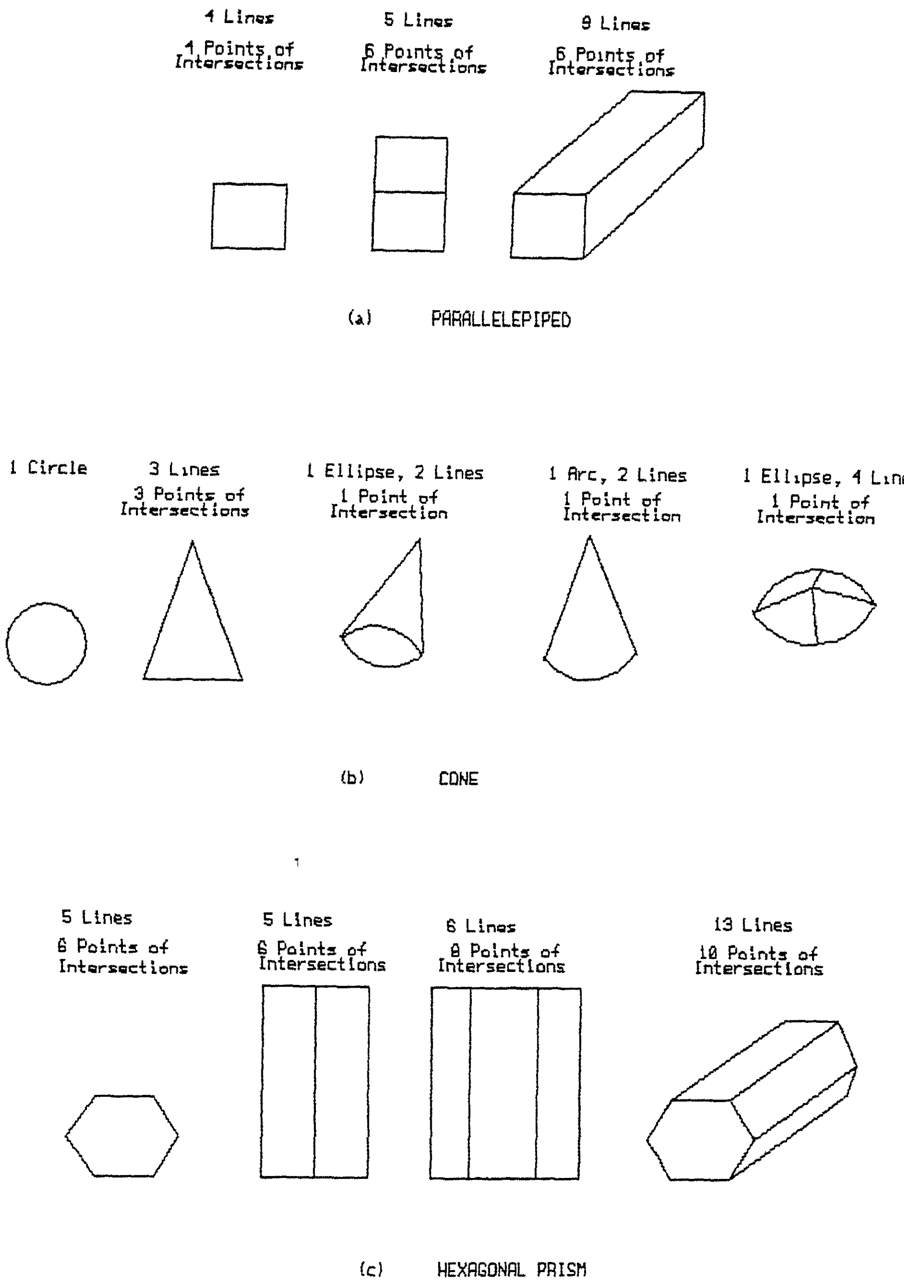

Figure 6.2 TOPOLOGICALLY DISTINCT VIEWS OF SIMPLE SHAPES

there is no need to enter those views in the visual potential of the assembly which contain the surface of the base which makes contact with the ground

The next step is to develop the interpretation technique using the above mentioned features and the model representation scheme Also with the modifications suggested above the visual potential of the object from now on will be called the *modified visual potential* in order to distinguish it from the visual potential suggested in [21]

6.3 DEVELOPING THE FEATURE INTERPRETATION SCHEME

As mentioned in section 5 2 3 the feature interpretation is carried out by either the decision tree approach or the pattern classification approach Both have their limitations and powers but with the kinds of new computer architecture coming up the pattern classification approach looks attractive A further improvement can be made in this technique by incorporating heuristics

Using the same terminology as given in section 5.2 3 let us denote the total number of features the vision system will look for as 'n'. Let's designate the unit vectors along each dimension represented by a feature with the following convention ·

Feature	Notation
Line	$\hat{l}$
Point of intersection	$\hat{p}$
Circle	$\hat{c}$
Ellipse	$\hat{e}$
Arbitrary Curve	$\hat{a}$

Any node in the modified visual potential of an object can be represented by a point in this 5 dimensional space Each point may also correspond to the nodes of the modified visual potential of several objects. Now let's say that the number of lines in the viewed object are 'v', number of points of intersections in the viewed object are 'w', number of circles are 'x', number of ellipses are 'y' and the number of arbitrary curves are 'z'. This object can be represented as a point, called as a *feature point*, as

$$\vec{q} = v\hat{l} + w.\hat{p} + x.\hat{c} + y.\hat{e} + z.\hat{a}$$

As mentioned earlier this point can also be called as the feature vector of the viewed object. The objective now is to find the point corresponding to the nodes of the modified visual potential of each object which is *nearest* to $\vec{q}$. There are two pathological cases where this nearest point approach may not work ·

(i) When the nearest point corresponds to the node representing several objects

(ii) When there are two or more points nearest to that of the object viewed If these points correspond to the nodes representing the

same model there is no ambiguity and the decision can be taken immediately But the situation becomes troublesome when these points correspond to the nodes representing different objects

The best way to cross such an impasse is to move the camera by a large angle around the object and repeat the above process. This conflict can also be resolved through statistical approximations but if a moving camera is available the former approach will give better results because no approximation is involved in the approach The camera movement may not be necessary always when a conflict occurs if some heuristics are introduced in the decision algorithm For example if the two points nearest to the viewed object correspond to a curved and a polyhedron object and if the feature vector of the object has a zero dimension along $\hat{c}$, $\hat{e}$ and $\hat{a}$ the object can not be curved even though the pattern classification algorithm may suggest so based on its nearest neighbor criterion.

A formal algorithm to demonstrate the usage of this technique is given below It is important to note that the complete identification process is carried out in two phases The first phase is the calibration or the training phase and the second phase is the identification phase The calibration phase implemented by calling the algorithm ***calib()*** has to be performed only once in the beginning of the whole process The identification phase is necessary when an object is to be identified once the system is calibrated to identify a set of object containing that object This is done by calling the algorithm ***identify()***. The algorithm calib() uses the algorithm ***represent()*** given in section 6 2 to generate the modified visual potential of all the objects that the vision

system is supposed to identify Also important is the fact that each such object is addressed by a unique ***object id***. Also each point in the feature space corresponding to a node of the modified visual potential of the object is addressed by a ***node point id***.

procedure calib()

begin

for each object generate its modified visual potential

begin

represent(),

end

for each node of the each modified visual potential generate a point in the feature space

begin

for the representation contained by the node put a point in the feature space ,

If (the node has got a link to any other node of any other modified visual potential)

begin

tag the point as 'plural' and store the object id no. of all the objects that correspond to the linked nodes ;

end

else

```
                    begin
                            tag the point as 'singular' ,
                    end
            end
return(),
end


procedure identify()
begin
        Generate the description of the object(s) in the scene
                    find out the no of lines (v), no  points of
                    intersection among them (w), no  of circles
                    (x), no. of ellipses (y), no  of arbitrary
                    curves (z) (each of length above a threshold
                     which minimises the detection of such
                     curves produced as a result of noise) ,
                    construct the feature point q⃗ for the
                     scene with the above data ,

        Find the nearest neighbor of the feature point
            upbound = δ ,
            min = ε ;
          for each node point in the feature space
          begin
                    If (the distance of the node point from the
```

```
                    feature point  is < min)
                         begin
                              store the node point id ,
                         end
     end


Find out the object(s) belonging to the node point ·
          If( node point tag = singular)
               begin
                    get the object id of the object associated
                    with the node point ;
                    return(object id);
               end
          else
               begin
                    apply heuristics if possible through the
                    data available from the feature vector in
                    order to delete as many objects as
                    possible from the list of probable
                    objects   ,
                    If( this list now contain only one object )
                         begin
                              return(object id),
                         end
                    else
```

```
                                        begin
                                               move the camera to a
                                               new orientation around
                                               the object ,
                                               take another image ;
                                               no_of_moves =
                                                        no_of_moves+1;

                                               If( no_of_moves >= count)
                                                  begin
                                                     report failure in
                                                     identification ;
                                                  end
                                               else
                                                  begin
                                                     identify() ,
                                                  end
                                        end
                        end
end
```

min is the variable used to store the minimum of the distance of all node points from the feature point ***upbound*** is the user selected upperbound for ***min***. ϵ is a large constant used to initialise the variable ***min*** and δ is

the user selected constant assigned to the *upbound*. δ can be large for the situations with little noise and occlusion but should be kept small for situations having a large amount of noise or occlusion.

Also the procedure *identify* is called recursively when the camera is moved and a new image is taken. A global counter designated by the ***no_of_moves*** should measure the level of recursions The constant ***count*** is selected depending on the complexity of the objects, noise, occlusion etc. Typically this value can be set to five. If the counter goes beyond this value the system reports a failure to identify the object Manual feedback of the identity in such a situation can be used by the system to train itself This is particularly helpful in occlusion situations. (Next section discusses this aspect further.)

6.4 THE STRENGTHS AND LIMITATIONS OF THE OF THE TOPOLOGICAL CONTOUR BASED MATCHING APPROACH

Sections 6 1, 6.2 and 6 3 elaborated on the basic details of the topological contour based matching approach for 3-D object recognition. This section discusses the strength and limitations of this approach by comparing its features with the characteristics of an ideal vision system as laid down in section 5 1 2.

1. If the nodes for the modified visual potential of the object (of complexity

of the order of that addressed in this work) corresponding to *all* topologically distinct views of the object can be generated, the model representation scheme suggested in section 6 2 essentially becomes view independent. This holds true for primitive objects such as nuts, bolts, washers, bearings etc As mentioned earlier this cannot be done through manual means for asymmetric objects such as a subassembly However with further research automatic generation of all such views possibly by accessing the solid model of the object from the CAD database, is possible which will relieve this from this limitations for such kinds of objects

2. The system can not take arbitrarily complicated objects but with the modified visual potential for the objects and its automatic generation the system adopting this technique will definitely perform better compared to existing methods More versatility can be brought in by incorporating more knowledge and heuristics about the object obtained through other sources This is the approach even human vision which possess such versatility, has adhered to

3. Though the identification algorithm can not by itself detect an object which is occluded by any other object, the presence of a moving camera will help segment the two objects better This means that the result of feature interpretation will provide a sort of feedback for the preprocessing algorithms which take part in segmentation of the occluded objects in the image.

even if the feature point of the viewed object does not exactly match any node of any object model, yet the system's tendency to find the nearest point leads it to carry out the identification. This operation with two or more views of the same object(s) obtained through the moving camera will result in reporting the type of the object with better confidence For example if the actual feature point of a view of an object is given as

$$\vec{q} = 5\,\hat{l} + 7\,\hat{p} + 3\,\hat{c} + 0\,\hat{e} + 0\,\hat{a}$$

and due to noise there is a line missing from the object's view the feature point becomes

$$\vec{q} = 4\,\hat{l} + 7\,\hat{p} + 3\,\hat{c} + 0\,\hat{e} + 0\,\hat{a}$$

With the nearest neighbor approach there is a good chance that the system will tolerate the absence of a line by still finding out the same nearest point The same will also hold true if there is a spurious line or any other spurious feature present Such small instances of noise can thus be easily tolerated. Yet in situations where there is considerably more noise, or if one needs better reliability it will be always useful to move the camera and take two or three views of the object before taking a the final decision

5. The speed of convergence to a decision will largely depend on how the features are extracted? This means which algorithm is used for detecting lines, their points of intersections, circles etc and whether they are implemented in hardware or software? For example Hough Transform is good for detecting lines but becomes too slow for arbitrary curves or even circles. Detecting circles or arbitrary curves can best be done through slope density function and so on Moreover, ideally each of these algorithms should

work in parallel on the image by assigning its implementation to a separate hardware unit The result of detection as reported by each of these units should be operated on by a separate unit. Such an implementation will quickly lead to the final result Pattern classification for relatively simpler objects can be carried out in software as it only involves finding the nearest neighbor and, therefore, will not consume too much time Software implementation of pattern classification is necessary if the system is to be trained for a particular kind of environment

6. If the visual potential of the object model can be generated using automatic means and if pattern classification is performed in software, modifying the system to handle new objects and new situations will be a fairly easy. However, till the automatic scheme of visual potential generation is developed it will be a tedious procedure to introduce new objects, especially if they are complex such as a sub-assembly of machine parts

7 Another attractive feature with the pattern classification scheme is that it can easily provide a quantitative estimate of the the 'belongingness' of a pattern to a class This number can be quickly calculated from the distance of the feature point of the object to its nearest neighbor. Thus if the feature point exactly matches a node of any modified visual potential and if that node corresponds to only one object, the system can declare with certainty that the object is the one corresponding to that node When two or three views of the object are taken through the moving camera this estimate can be the weighted average of the distance to the nearest neighbor. This weight should be

increased with the number of views taken as in each such view some objects are deleted from the list of possible objects

6.4 SUMMARY

A new technique has been proposed in this chapter for solving the feature interpretation problem in robot vision The motivation for proposing this technique was to use simple computations for feature extraction and interpretation taking advantage of a moving camera mounted on the robot arm. The emphasis was on extracting the features which represent a topological structure with minimum computation and to investigate the logic which gives such a structure. The potential of this technique which still requires further work was also discussed by comparing it with the characteristics an ideal vision system should possess The requirement of an edge enhanced image by the proposed technique can be considered as its shortcoming. However, with the algorithms for preprocessing fast going into the realm of hardware, this will no longer be true. Systems which do not require such preprocessing usually are even more computation intensive.

CHAPTER 7

CONCLUSION

A PC-AT has been chosen as the host machine for this project. Though convenient for the development work, it does not possess many of the features needed by a host machine of a robot vision system Its data I/O is slow to the extent that if even equiped with hardware tools for preprocesing it can not grab and process the data in real time A 68020 based system operating at 25 MHz is 'quite close' towards meeting this requirement Its long word I/O transfer mechanism and the availability of number of add-on cards on the standard VME bus to perform the preprocessing operations makes the system well suited to robot vision applications Coupled with this is the availablity of UNIX on these systems which helps standardise the interface with other machines.

The preprocessing should as far as possible be done in hardware. For example the 3x3 convolution needed to perform the edge detection over the 256 gray level 256x256 image will require time of the order of two to three minutes if performed on a PC-AT equiped with a floating point coprocessor Even on a 25 MHz 68020 system this will require around twenty to thirty seconds. Using add-on cards on the PC-AT for preprocessing can not improve the performance significantly as the time required for the PC to interact with these cards through the PC Bus is not less. A 68020 based system interacting with such cards through the VME bus can boost the system performance to near real time

Feature interpretation using the topological contour matching through pattern classification approach is a better solution over most of the existing methods This approach is especially suitable for robot vision problems employing a moving camera New architectures can be developed to exploit this technique in real time Such architectures would consist of two units one each for the feature extraction and feature interpretation. Designed to detect and/or measure the features of the type mentioned in chapter 6, the feature extraction unit will consist of sub-units one each for the each feature. The output of these sub-units would be combined through the feature interpretation unit which will perform pattern classification over these feature measurements Another attractive feature of this technique is that through further work the toplogically distinct views of an object can directly be generated through the engineering drawings of the object This means that

the integration of this technique with CAD databases will be fairly easy However, to make this technique powerful enough to deal with the general robot vision problems, further work needs to be done in the following directions

(i) To generate the topologically distinct views of an object of any complexity through automatic means from the object description available through CAD databases.

(ii) To reduce the diameter of the visual potential further by putting constraints and knowledge about the scene

APPENDIX A.1

PCB LAYOUT

For reliable and precise operation of the card, a four layer implementation is advisable Currently the proposed design has been tested with a two layer (viz the solder layer and the component layer) card Due to the limited space available on a card which is to be fitted in the PC expansion slot, the circuit needs a little portion to be implemented through another layer of jumpers The solder layer of the PCB has been shown in Figure A.1.1 and the component layer of the PCB has been shown in Figure A 1.2. The placement of chips and other components on the PCB has been shown in Figure A 1.3

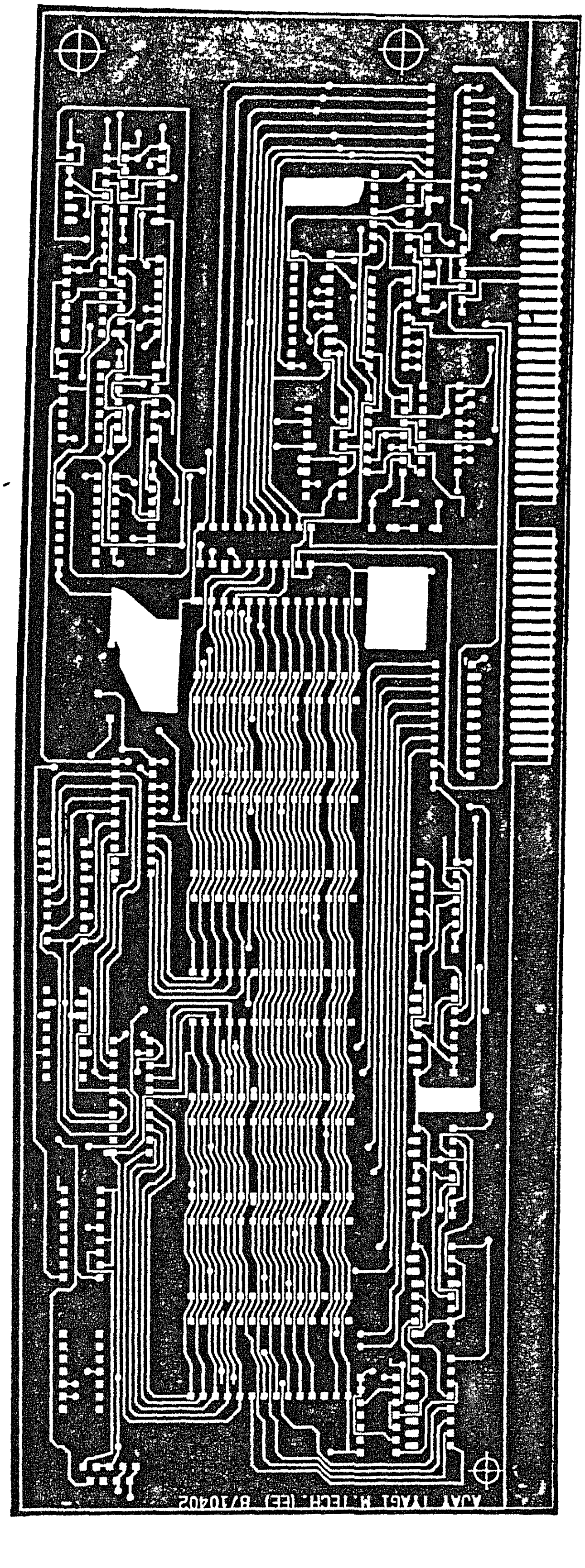

Figure A.1.1 PCB SOLDER SIDE

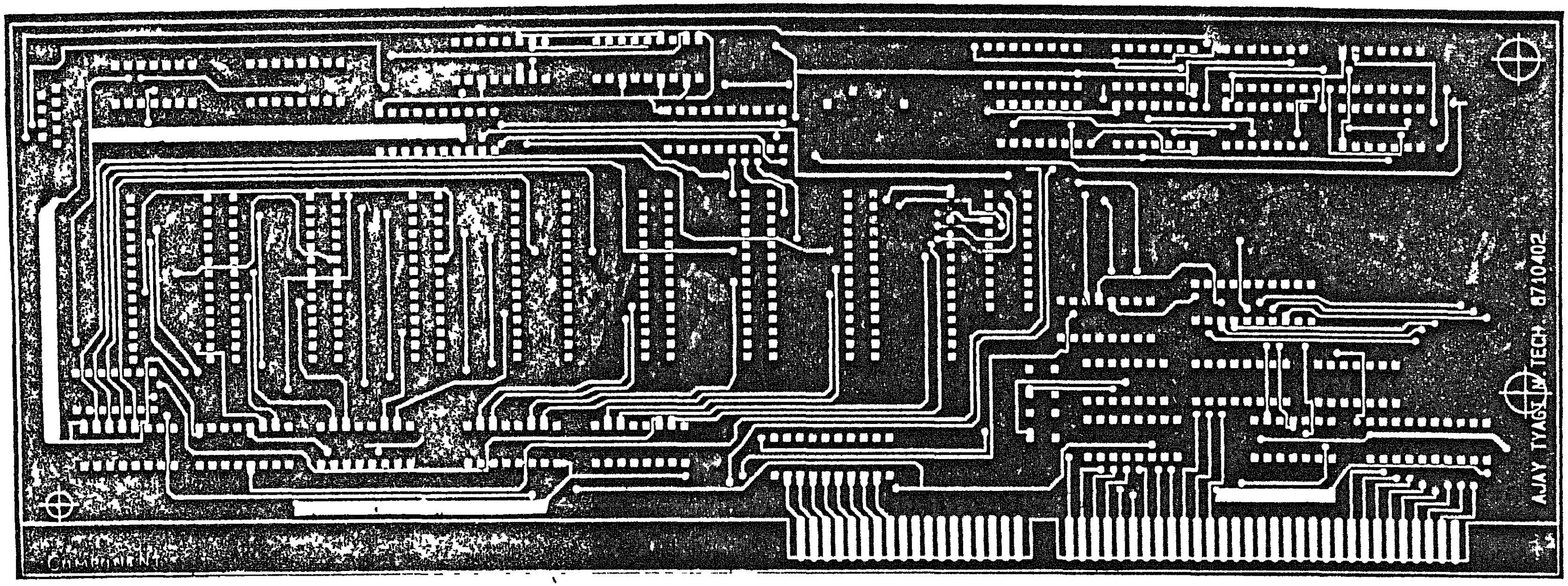

Figure A.1.2 PCB COMPONENT SIDE

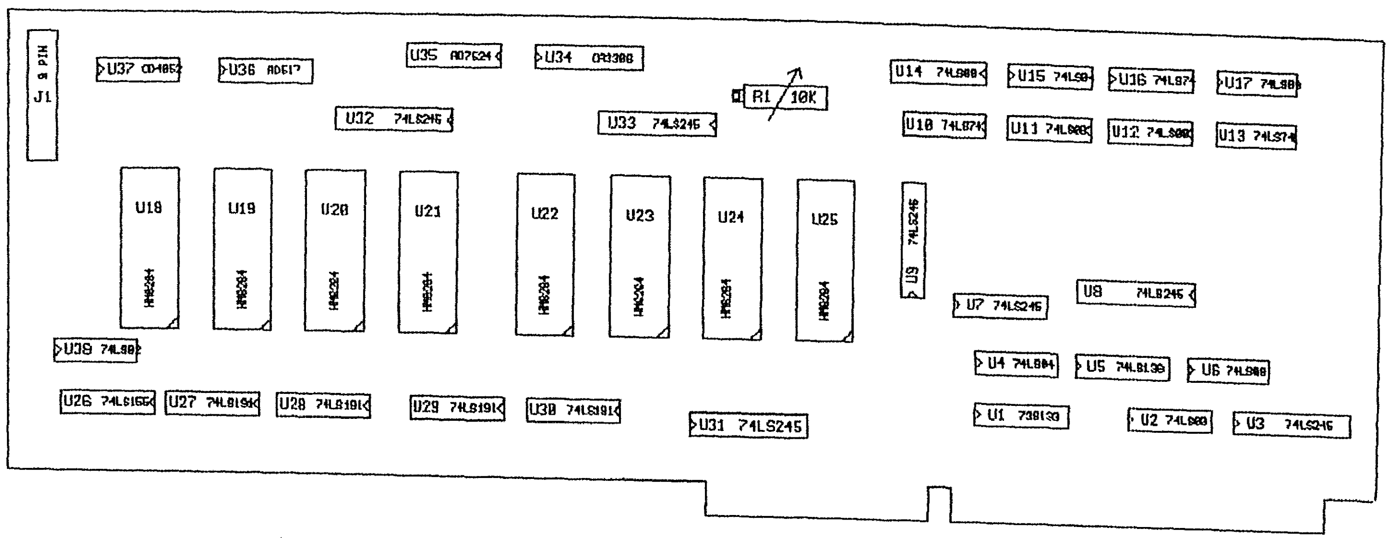

Figure A.1.3 PLACEMENT OF CHIPS

APPENDIX A2

DEVICE DRIVER LISTING

There are various ways of writing the device driver All these approaches will essentially be built upon the kernel suggested in Chapter 4 One approach is to provide five seperate programs, one each for each mode of operation of the card These programs will be executable from the command interpreter of DOS Such five programs have been given on the next pages The program for Initialising the card, Frame Capture and Frame Display will remain the same for both PC-AT and PC-XT However, the programs for Frame Input and Frame Output will be different for these two systems. Presently, programs for Frame Input and Frame Output have been given for PC-AT only For PC-XT, slight modification will have to be made and only the odd bank of the Frame Memory will be accessed thus giving an image resolution of 128x256

```
/* ===========================================================
                Program to initialise the card
=========================================================== */

#include <stdio.h>
#include <dos.h>
#include <stdlib.h>

#define init outportb(0x7cf,0x00)
#define fc outportb(0x7cf,0x01)
#define fd outportb(0x7cf,0x05)
#define fi outportb(0x7cf,0x07)
#define fo outportb(0x7cf,0x03)

main()
{

        init,           /* Initialising the card              */

}
```

```
/* ==============================================================
                  Program for Frame Capture
============================================================= */

#include <stdio.h>
#include <dos.h>
#include <stdlib.h>

#define init outportb(0x7cf,0x00)
#define fc outportb(0x7cf,0x01)
#define fd outportb(0x7cf,0x05)
#define fi outportb(0x7cf,0x07)
#define fo outportb(0x7cf,0x03)

main()
{
        unsigned char c,


        init; /* Initialising the card      */

        fc;   /* Putting the card in the frame capture mode  */

/* Polling the FA bit of the status port  */
        c=inportb(0x7cf);
        while( (c & 0x80) != 0x80)
                c=inportb(0x7cf);

        init,     /* Resting the card back       */

        printf("\n A FRAME HAS BEEN CAPTURED.. ."),
        printf("\n USE 'FI' OR 'FD' FOR INPUT OR DISPLAY...");
}
```

```
/* ============================================================
                  Program for Frame Display
=============================================================== */

#include <stdio.h>
#include <dos.h>
#include <stdlib h>

#define init outportb(0x7cf,0x00)
#define fc outportb(0x7cf,0x01)
#define fd outportb(0x7cf,0x05)
#define fi outportb(0x7cf,0x07)
#define fo outportb(0x7cf,0x03)

main()
{

        init,   /* Initialising the card            */


        fd;     /* Putting the card in the frame display mode  */

 }
```

```
/*==============================================================
                Program for Frame Input
            ---------------------------------

It uses the  control  register  and the status  register for
issuing commands for the five modes of operation of the card
and  enquiring  about the  status  of various signals on the
card  The address for both is 07CFh.

The 64K image data is read from the onboard memory of the card
into a file  specified  by the user through repeated input 32K
times from the port 03CFh
The 64K image data is  displayed  on  the monitor continuously
once  it is  written  into  the  onboard  memory from  a  file
specified  by the user through  repeated  output  64K times to
the port 03CFh.

============================================================== */

#include <stdio.h>
#include <dos h>
#include <stdlib.h>
#include <bios.h>
#include <ctype.h>
#include <fcntl.h>

#define init outportb(0x7cf,0x00)
#define fc outportb(0x7cf,0x01)
#define fd outportb(0x7cf,0x05)
#define fi outportb(0x7cf,0x07)
#define fo outportb(0x7cf,0x03)

main()
{
        unsigned int eurl,
        long i,j,
        char c,s[30];
        FILE *fopen(),*fp,




init,  /* Initialising the card          */

printf("\n Do you want the output to be stored in a file (y/n) ? ");

while(((c=getchar()) !='y') && (c !='Y') && (c !='n') && (c !='N'))
        printf("\n Give yes(y) or no(n) - ");
```

```
if(c == 'y' || c == 'Y')
{   printf("\n Give the file name --"),
    scanf("%s",s),
    fp=fopen(s,"w");   /* Creating the file     */

printf("\n Do you want the contents to be displayed on the screen (y/n)?");
while(((c = getchar()) != 'y') && (c != 'Y') && (c != 'n') && (c != 'N'))
    printf("\n Give yes(y) or no(n) - "),

        fi; /* Putting the card in the frame input mode  */
    printf("\n");

    for(i=0, i< 256; i++)    /* Line Counter      */
    {
        for(j=0, j< 128 ; j++)     /* Column Counter  */
        {
        /* Inputting the lower (odd) byte
           from the Frame Memory */
                          eur1=inport(0x3cf),
                          eur1= (eur1 & 0x3f);

                  /* Storing the lower (odd) byte     */
            fprintf(fp,"%d ",eur1);

        /* Displaying the lower(odd) byte    */
                if((c == 'y') || (c == 'Y'))
            {
                printf("%d ",eur1);
             }

        /* Inputting the upper (even) byte
           from the Frame Memory */
                          eur1=inport(0x3cf);
                          eur1= (eur1 & 0x3f00);
            eur1= ( eur1 >> 8) ,

                  /* Storing the upper (even) byte     */
            fprintf(fp,"%d ",eur1),

        /* Displaying the upper (even) byte  */
                if((c == 'y') || (c == 'Y'))
            {
                printf("%d ",eur1),
             }
        }

/* Dummy carriage return after each line          */

        fprintf(fp,"\n");
        if((c == 'y') || (c == 'Y'))
            printf("\n");
    }

    }
```

```
else
{
    printf("\n");

    fi;   /* Putting the card in the frame input mode    */

    for(i=0, i< 256, i++)    /* Line Counter          */
    {
        for(j=0, j< 128 ; j++)  /* Column Counter    */
        {
            /* Inputting the lower (odd) byte
           from the Frame Memory */
                            eur1=inport(0x3cf);
                            eur1= (eur1 & 0x3f),

        /* Displaying the lower (odd) byte */
            printf("%d ",eur1),

            /* Inputting the upper (even) byte
           from the Frame Memory */
                            eur1=inport(0x3cf);
                            eur1= (eur1 & 0x3f00),
            eur1= (eur1 >> 8);

        /* Displaying the upper (even) byte */
            printf("%d ",eur1);


        }
        /* Dummy carriage return after each line  */
        printf("\n");

    }
}

init,   /* Reseting the card back   */

}
```

```
/* ================================================================
                    Program for Frame Output
               -----------------------------

  It uses the  control  register  and the status  register for
  issuing commands for the five modes of operation of the card
  and  enquiring  about the  status  of various signals on the
  card. The address for both is 07CFh.

  The 64K image data is read from the onboard memory of the card
  into a file  specified  by the user through repeated input 32K
  times from the port 03CFh
  The 64K image data is  displayed  on  the monitor continuously
  once  it is  written  into  the  onboard  memory from  a  file
  specified  by the user through  repeated  output  64K times to
  the port 03CFh.

================================================================ */

#include <stdio h>
#include <dos.h>
#include <stdlib.h>
#include <bios.h>
#include <ctype.h>
#include <fcntl.h>

#define init outportb(0x7cf,0x00)
#define fc outportb(0x7cf,0x01)
#define fd outportb(0x7cf,0x05)
#define fi outportb(0x7cf,0x07)
#define fo outportb(0x7cf,0x03)

main()
{
        unsigned int eur1,eur2;
        long i,j;
        char c,s[30],
        FILE *fopen(),*fp;


init;  /* Initialising the card     */

/* Accepting the file name contaning the file to be displayed   */
        printf("\n Give the file name --");
        scanf("%s",s);

        fp=fopen(s,"r"),      /* Opening the file  */

printf("\nDo you want the contents to be displayed on the screen(y/n)?")
while(((c = getchar()) != 'y') && (c != 'Y') && (c != 'n') && (c != 'N')
        printf("\n Give yes(y) or no(n) - ");

        fo;       /*  Putting the card in the frame output mode  */
        printf("\n");
```

```
for(i=0, i< 256; i++)          /* Line counter      */
{
        for(j=0; j< 128 ; j++)  /* Column Counter     */
        {   /* Reading the lower (odd) byte from the file */

                fscanf(fp,"%s",s),
                eur1=atoi(s);

             /* outputing the byte in the Frame Memory
                Odd Bank                                */
                outport(0x3cf,eur1),

                if((c == 'y') || (c == 'Y'))
                {   /* Displaying the lower (odd) byte */

                        printf("%d ",eur1),

                 }

                c=getc(fp),  /* Dummy blank between two
                                bytes                  */

           /* Reading the upper (odd) byte from the file */

                fscanf(fp,"%s",s);
                eur2=atoi(s);

                eur2= (eur2 << 8);
                eur2= (eur1 | eur2),

             /* outputing the byte in the Frame Memory
                Odd Bank                                */
                outport(0x3cf,eur2);


                if((c == 'y') || (c == 'Y'))
                {   /* Displaying the upper (even) byte */

                        printf("%d ", (eur2 >> 8));
                 }
        }

        getc(fp), /* Dummy carriage return after one line */

        /* Displaying a carriage return after one line */
        if((c == 'y') || (c == 'Y'))
                printf("\n"),

}

init,    /*  Reseting the card back  */

}
```

APPENDIX A.3

SIMULATOR

A simulator can be built to generate synthetically the various views of a 3-D object, as one will see through a camera As part of its kernel such a simulator will use the fundamental theorems of Projective Geometry and algorithms of hidden line/hidden surface removal adopted from Computer Graphics

The fundamental theorems of projective geometry have been explained through rigorous mathemtics in the available literature [10] This is also true with hidden line/surface algorithm for polyhedra or an object composed of polyhedra The problem for hidden line/surface removal for curved objects is quite difficult and has received adhoc mathematical treatment [22] In other words developing a simulator for curved objects is a research problem in itself Furthermore, the simulator may generate silhouette and/or shaded

images. The monitors currently available on PC-AT (e g CGA, EGA etc.) do not provide shaded graphics Thus only silhouette images can be generated on a PC-AT This is acceptable in this work as the 3-D object recognition technique proposed in Chapter 6 only requires the silhouette of the various views of an object. It is necessary however that the simulator provides different views of the same object in order to simulate the moving camera effect also

An outline of the fundamental theory behind such a simulator is given below This will be followed by a brief explanation of how such a simulator can be used to test vision algorithms in general and the one proposed in this thesis in particular.

A.3.1 FUNDAMENTAL THEORY

The objective of such a simulator should be to provide a 256x256 image (or a 512x512 image, as the vision algorithm demands) As shown in Figure A.3.1 such an image can be generated by projecting the object model through the camera lens system onto to the image plane The complete projection system resides in a world coordinate system The origin of this coordinate system is located at center of the image plane

For polyhedron objects, the object model has to be stored in the form of the 3-D coordinates of a reference corner and its various other corners, a list of lines among them, and a list of planes formed by the set of such lines

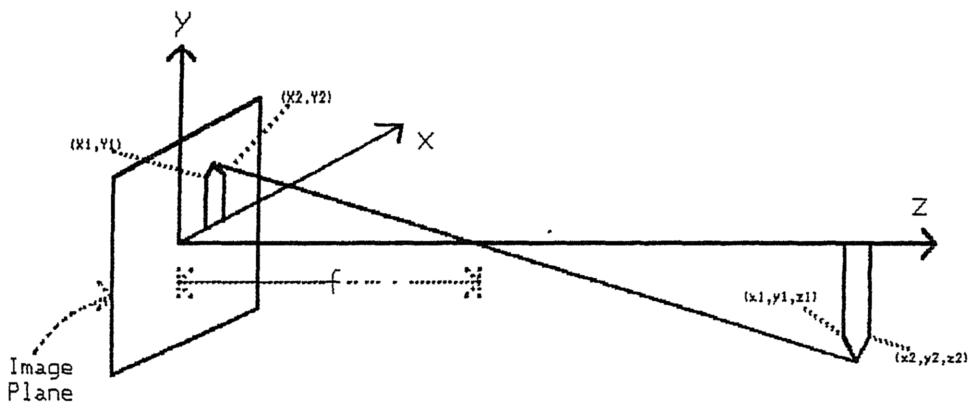

Figure A.3.1 Camera Model

For curved objects the object model has to be stored in the form of the coordinates of a reference corner and of other corners if any, a list of lines among them if any, a list of arcs through them, and a list of planes formed by the set of such arcs or lines Thus for generating silhouette of curved object like a cone the description may become too complex But as the 3-D object recognition to be tested using this simulator only requires the outer profiles of a curved object (as obtained through edge detection) a compromise can be made by deleting out all arcs and lines except those which represent the outer profile of the object.

Moving camera effect can be simulated using the 3-D transformation algorithms on these object models An object moved along the x-axis by -10 units is equivalent to the camera being moved by +10 units along the x-axis or a 20° rotation of the object along the y-axis is tantamount to the camera being rotated by -20° along the y-axis

A.3.1.1 ***Projection Equation*** : The projection of an object can be modeled through the direct application of perspective transformation on each corner point of the object As the lines and curves between two points remain intact after such a transformation the list of lines and the list of arcs can be used to generate the lines and curves for the 2-D image plane of the camera Thus if the two corner points of the object have cordinates **x1,y1,z1** and **x2,y2,z2** in the 3-D world coordinate system and if the camera has a focal length **f** the 2-D coordinates of these two points in the camera image plane will be

$$X1 = -(x1*f)/(z1-f) \quad , \quad Y1 = -(y1*f)/(z1-f) \; ;$$

$$X2 = -(x2*f)/(z2-f) \quad , \quad Y2 = -(y2*f)/(z2-f) ;$$

If there is a line between (x1,y1,z1) and (x2,y2,z2) or a curve passing through these two points in the 3-D world coordinate system there would be a line between (X1,Y1) and (X2,Y2) or a curve passing through in the 2-D image plane also However not all lines and arcs of an object will be visible from any eyepoint Thus before drawing the various lines and curves present among all such corner points of an object in order to produce the projected view of the object, it is necessary to apply hidden line and hidden surface algorithm to turn the attribute of such lines and curves as 'hidden' The best algorithm applicable for such a kind of problem is the Roberts Algorithm [22] It is applicable to polyhedon objects only and some modification will be needed before applying it to the curves objects All such algorithms, however, have to have an apriori knowledge about the types of object present in the scene. Therfore it is important to consider how such objects are represented This has been expalined below through the examples of a polyhedron and a curved object

A.3.1.2 *Represention for a Parallelopied* : A parallelopiped can be represented by a refence corner point and the coordinates of its other seven corner points These eight points will have to be stored in a list of points with each point accessed through an index in the list An edge of the parallelopiped can be representd by a structure containing the pair of indices of the two points forming the edge, and the attribute of the edge i e 'hidden' or 'visible' The list of lines for each parallelopiped [illegible] of twelve rows corresponding to each such edge of the parallelopiped A plane of

the parallelopiped can be represented by a structure consisting of the indices of the four points (in the list of points) forming a plane of the object. the list of planes for each parallelopiped will consist of eight rows coressponding to the eight planes of the parallelopiped

With such a representation the Roberts hidden line hidden surface algorithm can be directly applied. It is necessary to have the z-coordinate of each corner of the object ***after*** the application of perpective transformation Though this coordinate of the object is not necessary for producing the 2-D projection on a display it is a key requirement for the Roberts Algorithm For a corner of the object having coordinate **x,y,z** the z cordinate can be obtained using the perspective transformation equation as

$$Z= -(z*f)/(z-f)$$

For the sake of avoiding any confusion, all the coordinates of all points of the object can be assumed as belonging to a $2\frac{1}{2}$ space while applying the Roberts Algorithm over the object.

A 3.1.3 ***Representing a Cone*** A cone can be repreented by a set of six points, four arcs and four lines and one plane These points correspond to the center of the base, the four corner points along the periphery of the base (two on the line joining the center of the base with the eyepoint and two on the line perpendicular to it) and the apex of the cone

The four arcs correspond to the four possible set of three corner points of the base The four lines corresepond to the four edges joining the points on the periphery of the base with the apex The plane is the base

plane of the cone. Though such a description contains redundancy, it is necessary to keep this in order to into account the rotation of the cone along an arbitrary axis

With this description the cone can be represented by a list of points a list of lines, a list of arcs and a structure for the plane, in the same manner as given for the parallelepiped Roberts Algorithm can also be applied in the similar manner to find out the attribute of the four lines and the plane Once this is accomplished application of simple 3-D geometrical theorems can lead to find which of the two arcs are visible and which are hidden

A.3.2 USING THE SIMULATOR TO TEST VISION ALGORITHMS

Using this kind of a simulator for testing vision algorithms is straightforward The image plane of the camera can be considered as a clipping plane of resolution 256x256 which clips the projected view of any object The clipping problem, however, is not trivial and demands sophisticated algorithms [22] The image received by this plane can be used by the vision algorithm directly If the vision algorithm demands a movement of the camera the simulator can move the object in the scene in opposite direction through appropriate transformation and project the object again on the image plane. The image plane will again clip the projected view to give the vision algorithm the new view of the object

To generate a view of a complex object consisting of such primitive

objects or a scene consisting of multiple numbers of such objects require further investigation in hidden line and hidden surface algorithms. If coupled with the ability to generate the shaded view of these objects, this simulator can become a vital tool for experimentation with vision algorithms

REFERENCES

1 D.H. Ballard & C.M Brown "Computer Vision" Prentice Hall, Englewood Cliffs, New Jersy 1982

2. D Nitzan "Three Dimensional Vision Structure for Robot Applications" IEEE Trans. on Pattern Analysis & Machine Intelligence, vol. 10, no. 3, pp 291-309, May 1988

3 A C Kak & J.S Albus "Sensors for Intelligent Robots" Handbook Industrial Robotics Shimon y NOF (ed.) John Wiley & Sons, NewYork 1985

4. J L. C Sanz "Introduction to PAMI Special Issue on Industrial Machine Vision and Computer Vision Technology - Part II" IEEE Trans. on Pattern Analysis & Machine Intelligence, vol 10, no 3, May 1988

5 R Gulati "Monochrome and Color Television" Wiley Eastern Ltd., New Delhi 1987.

6 "PC-AT Technical Reference" IBM Corporation, 1984

7 "CMOS Integrated Circuits" RCA Corporation, 1983

8 "Data Conversion Products HandBook" Analog Devices, Inc , 1988

9 P J Besl & R C Jain "Three Dimensional Object Recognition" ACM Computing Surveys, vol 17, no 1, pp 75-145, March 1985

10 D. F. Rogers & J. A. Adams "Mathematical Elements for Computer Graphics" McGraw Hill Book Company, New York 1976

11. B. K. P. Horn "Robot Vision" MIT press, McGraw Hill Book Company, NewYork 1986

12 Special Issue on Artificial Neurons, IEEE , Computer, March 1988

13 D. J. Meagher "Geometric Modeling Using Octree Encoding"Computer Graphics & Image Processing, vol 19, no. 2, pp 129-147, June 1981

14 B I. Soroka & R K Bajcsy "A Program for Describing Complex Three Dimensional Solids Using Generalised Cylinders as Primitives" Proceedings of Pattern Recognition and Image Processing Conference (Chicago, Ill.) IEEE, pp 331-339, 1978

15 I Chakravarty & H. Freeman "Characteristic Views as Basis for Three Dimensional Object Recognition" Proceedings of the Society of Photo - Optical Instrumentation Engineers Conference on Robot Vision, vol 336 (Arlington, Va.), SPIE, Bellingham, Wash , pp 37 - 45, May 6-7, 1982

16 T C Henderson "Effecient 3-D Object Representation for Industrial Vision System" IEEE Trans. on Pattern Analysis & Machine Intelligence, vol. 5, no. 6, pp 609-617, Nov 1983

17. R D Duda & P E. Hart "Use of Hough Transformation to Detect Lines and Curves in Pictures" Communication ACM, vol. 15. no.1, pp 11-15, 1972

18 P.J Nahim "The Theory of Measurement of a Silhouette Description for Image Proccesing and Recognition" Pattern Recognition, vol 6, no.2, pp 85-95, 1974.

19 R D. Duda & P.E. Hart "Pattern Classification and Scene Analysis" John Wiley & Sons, NewYork 1983

20. C. I. Connolly "The Determination of Next Best View" Proceedings of

the International Conference on Robotics (St Louis Mo.) IEEE, pp 25-32, May 25-28, 1984

21 J. J. Koenderink and A. J Van Doorn "Internal Repersentation of Solid Shapes with Respect to Vision" Biological Cybernectics, vol. 32, no 4, pp 211-216, 1979

22 D F. Rogers "Procedural Elements for Computer Graphics" McGraw Hill International, New York 1985

EE-1989-M-TYA-IMA